Universidad
Carlos III de Madrid

ESCUELA POLITÉCNICA SUPERIOR

Grado en Ingeniería en Tecnologías Industriales

Departamento de Ingeniería de Sistemas y Automática

Trabajo de Fin de Grado

RGB-D SCAN MATCHING BASADO EN COVARIANCE MATRIX ADAPTATION - EVOLUTION STRATEGY

Autora:

María Granda Guerrero

Tutor del proyecto:

Fernando Martín Monar

Septiembre, 2016

RESUMEN

En el campo de la robótica, la búsqueda de algoritmos y métodos que permitan crear mapas robustos es uno de los temas más estudiados en los últimos años. Cada vez es mayor el interés en desarrollar robots autónomos, con el fin de emplearlos para tareas difíciles o que no pueden ser realizadas por los humanos, como por ejemplo exploraciones espaciales u operaciones de rescate. Para ello, es necesario que el robot sea capaz de enfrentarse a un entorno desconocido, y localizarse dentro de él.

Así surge el problema del SLAM (*Simultaneous Localization and Mapping*), que consiste en que el robot sea capaz de ir construyendo un mapa de un entorno desconocido, y a la vez ubicarse dentro de él. Un aspecto muy ligado a este problema es el *scan matching*, objeto principal de este trabajo. Con el *scan matching* se busca encontrar la transformación rígida (traslación y rotación) que alinea dos barridos del entorno diferentes. Estos barridos son proporcionados por sensores como cámaras RGB.

A lo largo de los años, se han ido buscando nuevas técnicas con las que poder realizar el *matching*. Algunas de estas técnicas son las denominadas Estrategias Evolutivas, mediante las que se busca resolver problemas de optimización basándose en los procesos de la evolución natural. En estas técnicas, se tiene una población inicial que evoluciona y varía de acuerdo al valor de coste obtenido hasta que converge a una solución.

En este trabajo se ha implementado una solución al *scan matching* basada en el *Covariance Matrix Adaptation-Evolution Strategy*. Con este método se busca realizar el *matching* entre dos *scans* minimizando una función de coste. Además se utilizan las propiedades del color para seleccionar los puntos característicos de cada barrido, reduciéndose el coste computacional del método.

ABSTRACT

In the robotics field, the research of new algorithms and methods that allow creating robust maps is one of the most studied issues during the last years. Interest in developing autonomous robots is growing further, in order to use them for difficult tasks or other tasks that can not be done by humans, as for example space explorations or rescue operations. To do that, it is necessary that the robot has the capacity of dealing with an unknown environment and of being located itself in the map.

For that matter SLAM problem appears (*Simultaneous Localization and Mapping*), which consists of the fact that the robot is capable of building a map of an unknown environment, and simultaneously of locating itself inside this environment. A highly related aspect to this problem is *scan matching*, which is the main subject of this project. *Scan matching*'s purpose is finding the rigid transformation (translation and rotation) that aligns two different scans of the environment. These scans are provided by sensors such as RGB cameras.

Throughout the years, researchers have been looking for new techniques that could perform the matching. Some of these methods are the so called *Evolutionary Strategies*, by means of which it is look to solve optimization problems based on the process of natural evolution. In these techniques, there is an initial population that evolves and changes according to the cost value obtained until it converges to a solution.

In this work, a solution of the *Scan Matching* problem has been implemented based on the *Covariance Matrix Adaptation-Evolution Strategy* algorithm. With this method it is look to achieve the matching between two scans by minimizing a cost function. Color properties are also used to select featured points in the scans, reducing the computational cost of the method.

Universidad Carlos III de Madrid

Índice de contenidos

Índice de figuras

Índice de acrónimos

(SLAM) *Simultaneous Localization and Mapping*

(DE) *Differential Evolution*

(CMA-ES) *Covariance Matrix Adaptation-Evolution Strategy*

(AE) *Algoritmos Evolutivos*

(CE) *Computación Evolutiva*

(PE) *Programación Evolutiva*

(AG) *Algoritmos Genéticos*

(ES) *Estrategias Evolutivas, del inglés, Evolutionary Strategies o Evolution Strategy*

(ICP) *Iterative Closest Point*

(ICL) *Iterative Closest Line*

(IMRP) *Iterative Matching Range Point*

(IDC) *Iterative Dual Correspondence*

(PSM) *Polar Scan Matching*

(CRF) *Conditionals Random Field*

(NDT) *Normal Distribution Transform*

(CIE) *Commission Internationale de l'Éclariage*

(JND) *Just Noticeable Difference*

Índice de tablas

Capítulo 1.

Introducción

Mapa se entiende como una representación métrica y gráfica de un entorno determinado. Los mapas más antiguos conocidos datan de 2300 a.C. Fueron elaborados por los babilonios sobre tablillas de arcilla, y su principal función era representar mediciones de tierras con el fin de cobrar impuestos. Diferentes civilizaciones del mundo, pertenecientes a diferentes épocas realizaron mapas para reflejar sus territorios conquistados, el conjunto de islas en el que habitaban, etc. Podemos decir que la humanidad, desde los inicios de la civilización, ha buscado la manera de plasmar el mundo [1].

En el campo de la robótica, la creación de mapas es uno de los temas más estudiados. Podemos encontrar diversos tipos de mapas: en función de la finalidad que tenga el mapa; si el entorno es interior o exterior, dinámico o no; etc. Podemos encontrar dos modelos principales: mapas geométricos y mapas topológicos. Los mapas geométricos buscan conseguir representaciones exactas y precisas del entorno. Una posible forma de representar estos mapas es como una rejilla de probabilidades, en la que la intensidad de cada celda equivale a la probabilidad de que esa región del entorno esté vacía. Los mapas topológicos son representaciones cualitativas del entorno, en los que aparecen elementos distintivos que pueden ser reconocidos con facilidad por el robot [2].

Analizando el entorno socioeconómico, actualmente la robótica es muy relevante y está presente cada vez en un mayor número de ámbitos. Desde la Revolución Industrial, se ha buscado la manera de introducir sistemas automatizados que realicen un elevado número de tareas en el ámbito productivo. A día de hoy, casi todos los procesos de fabricación son realizados de manera automatizada, empleando diferentes robots. Un claro ejemplo es el uso de brazos robóticos en el ensamblaje de las piezas de un automóvil. Su inclusión en el ámbito industrial permite realizar tareas con una mayor precisión, como la necesaria para la fabricación de un microchip, con mayor rapidez y con un menor coste. Esto ha tenido una alta repercusión en la sociedad en varios aspectos. Por un lado, se ha aumentado la seguridad del trabajador en su puesto al disminuir el número de tareas que podían resultar duras, tediosas o peligrosas, al haberlos sustituido por

robots. Y por otro lado, es posible obtener productos de una mayor calidad, a un precio menor.

Gracias a los avances en tecnología, cada vez son más los ámbitos en los que es posible emplear robots. Estos tienen la finalidad de ayudar a las personas, bien sea quitándoles de realizar ciertas tareas, como el robot aspirador en el ámbito doméstico; como de servir de apoyo y contribuir a realizarlas. Un ámbito en el que su uso presenta muchas ventajas es el de la medicina, y cuyo uso está cada vez más extendido. Los robots se utilizan en labores de traslado de muestras o distribución de medicamentos, disminuyendo el número de errores cometidos; en procesos de rehabilitación de pacientes; o incluso en operaciones. El uso de robots quirúrgicos presenta grandes ventajas, entre las que cabe destacar la precisión con la que se realiza, eliminando los posibles temblores de los médicos ante situaciones de tensión. Además, permiten acceder a zonas más complejas, disminuyéndose también el riesgo de dañar otras zonas.

Para la realización de muchas tareas tediosas, o que conllevan tener que transportar elementos pesados, exploraciones espaciales o submarinas, o incluso en trabajos de rescate, se intenta cada vez más introducir robots móviles, que dispongan de un elevado grado de autonomía. Para poder introducirlos en estos ámbitos, es necesario que el robot sea capaz de "enfrentarse" al entorno que le rodea. Uno de los principales objetivos de un robot móvil y/o autónomo es que tenga la capacidad de crear representaciones del entorno que le rodea, así como que sea capaz de encontrarse dentro de ese mismo mapa. Este problema se denomina *Simultaneous Localization and Mapping* (SLAM). Una de las principales dificultades del SLAM es que cualquier error cometido en las estimaciones de la posición, repercutirá en la construcción del mapa y, por lo tanto, llevará a obtener una localización incorrecta.

El *scan matching* es uno de los problemas principales del *mapping* (mapeo), y consiste en calcular la relación métrica o posición relativa (rotación y traslación) que existe entre dos barridos o *scans.* Estos *scans* se obtienen por medio de sensores, como sónar para espacios acuáticos, escáneres láser, sensores de visión, etc. El *RGB-D Scan Matching* se basa en utilizar sensores RGB-D que proporcionan información de profundidad y color en sus escaneos. En la *Figura 1* se muestra una explicación esquemática del funcionamiento del algoritmo.

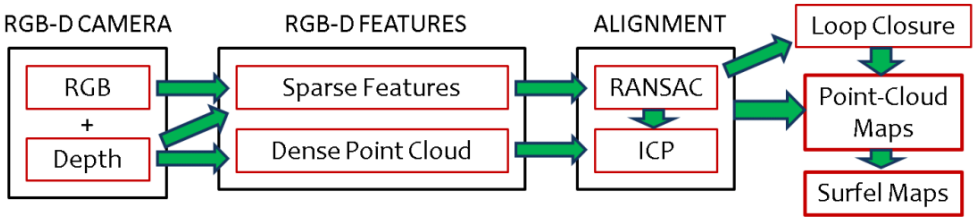

Figura 1. Esquema del algoritmo RGB-D Scan Matching [3]

En el presente trabajo se presenta una solución alternativa para la realización del *matching*, implementando un algoritmo perteneciente a la categoría de computación evolutiva: un algoritmo de estrategia evolutiva (*Evolution Strategy o Evolutionary Strategies*). A la categoría de computación evolutiva pertenecen también los algoritmos genéticos y la programación evolutiva, y todos ellos se estructuran de la siguiente manera: inicialización, mutación, recombinación y selección. Una descripción más detallada de estos algoritmos se realiza en el *Capítulo 4*.

1.1. Motivación y objetivos

Como se ha mencionado anteriormente, la creación de mapas robustos es uno de los temas más estudiados de la robótica. Los investigadores de este campo están en constante búsqueda de nuevos algoritmos o métodos que permitan la construcción de mejores mapas, más precisos; así como la utilización de nuevos sensores. Utilizando estos sensores, el robot puede obtener diferente información en función del tipo de sensor: si son externos, la información será espacial sobre el entorno; si en cambio son internos, será sobre variaciones en la posición del robot. Un avance en la creación de mapas ha sido el uso de cámaras RGB-D, mediante las cuales se puede obtener información del color y la profundidad del entorno que se está escaneando.

Para la conformación de estos mapas se emplean métodos como el *scan matching*, que consiste en buscar la traslación y rotación que existe entre dos scans a través de los puntos comunes, y poder combinarlos. Esta combinación o alineación puede llevarse a cabo utilizando diversos algoritmos, como estrategias evolutivas o algoritmos iterativos.

Este trabajo se ha basado en un trabajo previo realizado por Fernando Martín *et al.* [4], en el que se busca obtener la transformación que minimiza la distancia entre dos nubes de puntos utilizando un método evolutivo conocido como *Differential Evolution* (DE). Partiendo de él, el objetivo principal del trabajo

es implementar una nueva solución al problema del *matching* utilizando un algoritmo diferente, con el que se pretende obtener mejores resultados en la alineación de los *scans* que con los algoritmos ya implementados. En este caso, la solución buscada e implementada también pertenece al grupo de las Estrategias Evolutivas, el *Covariance Matrix Adaptation-Evolution Strategy* (CMA-ES).

Las Estrategias Evolutivas, son un conjunto de algoritmos pertenecientes al campo de la Computación Evolutiva (CE). Los estudios de la CE tienen como finalidad encontrar la manera de imitar los procesos naturales de evolución para resolver problemas de optimización. En las Estrategias Evolutivas, se parte de una población inicial formada por un número λ de individuos, que puede ser conocida o generada aleatoriamente según el problema. Para que el algoritmo avance, esta población va sufriendo diversas modificaciones o mutaciones en sus elementos. Para encontrar a los mejores individuos de la población, se utiliza una función de coste que se debe minimizar. De este modo, los elementos que tengan un menor valor de coste, serán empleados para obtener la nueva generación de individuos.

En el caso del método de este trabajo, el CMA-ES, estas mutaciones se obtienen a partir de una distribución normal multivariante. Para la obtención de la distribución es necesario el cálculo de dos parámetros principales: la media y la matriz de covarianza. Se parte de una población inicial calculada aleatoriamente, a la que se le aplica la función de coste. En función de dicho valor, se escogen los μ mejores elementos, es decir, aquellos que tienen un menor coste. Con ellos, se calcula un valor inicial de la media ponderada de la población. La estimación de la matriz inicial se realiza a través de una ponderación de los mejores individuos y la media inicial calculada anteriormente. Con estos valores de media y matriz de covarianza se obtiene la siguiente generación de la población con una distribución normal multivariante. Este proceso se repite hasta un número máximo de iteraciones. La finalidad de escoger los mejores individuos es ir obteniendo mejores poblaciones para ir aproximando los resultados a la solución óptima.

1.2 Herramientas utilizadas

Para la realización de este trabajo ha sido necesario emplear un ordenador *Lenovo Z50-70* con procesador *Intel Core i5* y sistema operativo *Windows 10*, en el que se ha instalado la herramienta MATLAB (*MATrix LABoratory*) para el desarrollo del algoritmo. Esta herramienta es un software matemático muy potente que cuenta con su propio lenguaje de programación y una extensa librería de funciones propias que facilitan el uso de la herramienta. Algunas de estas funciones son: manipulación de matrices (cálculo de la matriz inversa, el

determinante, etc.); operaciones con complejos; resolución de ecuaciones; etc., así como funciones visualización de datos en 2D y 3D. Esta última característica es uno de los motivos por los que esta herramienta es adecuada para la realización de este trabajo.

La finalidad del algoritmo es la realización de un *matching* entre dos imágenes. Para poder comprobar los resultados que se obtienen con el algoritmo implementado, se ha utilizado una base de datos compuesta por *scans* que contienen información de color y profundidad de un entorno real. En la actualidad, existen diferentes sensores con la capacidad de proporcionar barridos que contengan la información requerida para este trabajo. Particularmente en este caso, se ha utilizado el sensor *Microsoft Kinect*.

1.3 Estructura del documento

A continuación, se va a realizar una breve descripción de los apartados que conforman este trabajo:

- Capítulo 2. En este capítulo se realiza un pequeño análisis del estado del arte, para ir introduciendo al lector en el tema que se trata en este trabajo.
- Capítulo 3. Este capítulo consta de una descripción de los conceptos teóricos del *Scan matching*, y del uso del color en este método. También se explican en más detalle algunos algoritmos utilizados en *Scan matching*.
- Capítulo 4. Al igual que en el capítulo anterior, en este capítulo se exponen los conceptos teóricos de los algoritmos evolutivos, y más concretamente del *Covariance Matrix Adaptation-Evolution Strategy* (CMA-ES).
- Capítulo 5. En este capítulo se explica la solución implementada del algoritmo *CMA-ES Scan Matching*
- Capítulo 6. En este capítulo se exponen los resultados obtenidos, así como las diferentes variaciones realizadas en el algoritmo con el fin de mejorar los resultados.
- Capítulo 7. En este último capítulo se exponen las conclusiones finales del trabajo.

Capítulo 2.

Estado del arte

El campo de la robótica siempre ha presentado muchos retos. Uno de los más estudiados es la creación planos o mapas (mapeado, del inglés *mapping*) robustos, de manera que puedan ser utilizados, por ejemplo, para navegación, localización y/o posicionamiento de robots en diferentes entornos o telepresencia. Todo ello es posible gracias a los sensores, que proporcionan al robot información sobre el medio que le rodea.

Analizando los diferentes métodos de *scan matching*, encontramos tanto técnicas 2D [5,6], como 3D [7]. En algunos casos y con el fin de simplificar el problema se han realizado trabajos previos en 2D, buscando facilitar el análisis de datos, y posteriormente se realizan las correspondientes modificaciones para su uso en 3D [8].

Es posible distinguir también entre métodos locales [9] y globales [10] de *scan matching*. En el caso de los métodos locales, se combinan dos *scans,* siendo necesario partir de una posición inicial; mientras que en los métodos globales el *scan* actual es combinado con el modelo global, sin ser preciso que se proporcione una posición inicial.

Otra manera de clasificarlos es basándose en el método de asociación entre *scans* [4,11]:

- Basado en características (*feature to feature*), en cuyo caso es necesario obtener previamente una característica del entorno.
- Basado en puntos (*point to point*). A diferencia del anterior, no necesita que el entorno contenga ninguna estructura ni característica determinada.
- Mixto (*point to feature),* busca la correspondencia entre características y puntos.

Para resolver el problema de la optimización, el método más utilizado en *scan matching* es el algoritmo *Iterative Closest Point* (ICP) propuesto por Besl y McKay [12] en 1992, un método iterativo que busca minimizar la distancia entre dos *scans,* siendo necesaria una posición inicial estimada. Este recibe dos nubes de

puntos, y una estimación inicial de la traslación y la rotación, y el criterio para llegar al final de las iteraciones, la condición de fin. Los puntos de la primera nube (*source* o fuente) son combinados con el *vecino más cercano*[1] de la segunda nube (nube de referencia) de manera que se encuentre la transformación que minimiza la distancia entre los puntos. Debido a que es un método relativamente sencillo y con un coste computacional bajo, es posible utilizarlo en tiempo real [3, 4]. En la *Figura 2* se muestra el diagrama de flujo.

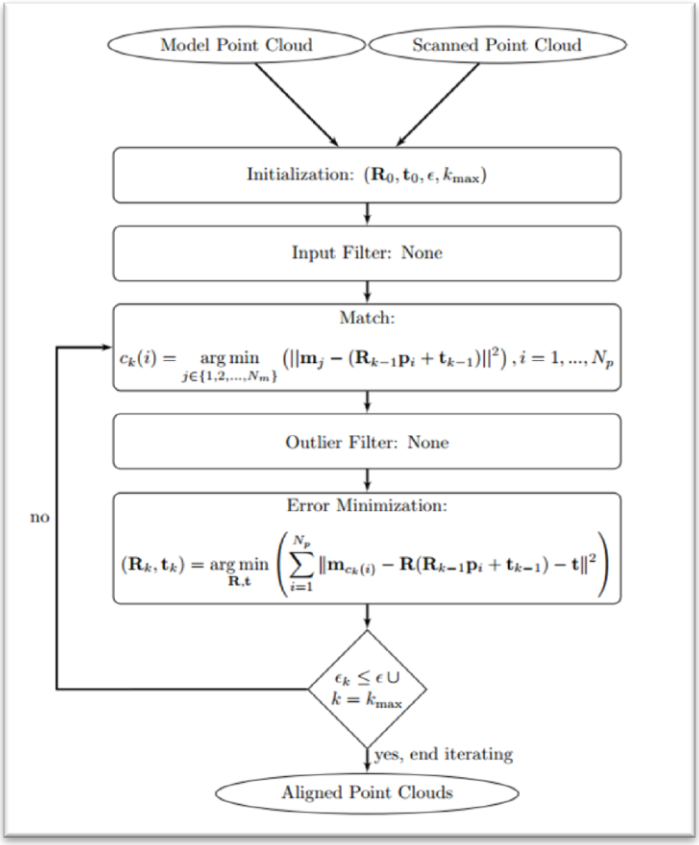

Figura 2. Diagrama de flujo del algoritmo ICP estándar [13]

Desde entonces, diferentes grupos de investigadores han ido desarrollando versiones de este algoritmo. Algunos autores como Rusinkiewicz *et al.* en 2001 [14] o Pomerleau *et al.* en 2010 [13] han realizado recopilaciones en las que se comparan algunas de las variaciones de este algoritmo.

Lu y Milios [9] proponen dos métodos de *scan matching* basados en el ICP. El primero de ellos, *Iterative Matching Range Point* (IMRP), trata la rotación y la traslación por separado, y utiliza parámetros para controlar la máxima rotación y traslación. Este método presenta muy buenos resultados para el caso de la rotación, ya que converge más rápido que el ICP. El segundo método, denominado

[1] en inglés, *nearest neighbour*

Iterative Dual Correspondence (IDC), combina el ICP y el IMRP: el ICP calcula la traslación, mientras que el IMRP calcula la rotación. Diosi *et al.* [11] presentan un método, denominado *Polar Scan Matching* (PSM), que no necesita encontrar asociaciones entre puntos, simplemente se combinan aquellos puntos que tienen el mismo comportamiento (bearing). Se han desarrollado también diferentes ampliaciones del ICP, como por ejemplo *Color*-ICP [15], que utiliza colores y texturas de los objetos como características; o *Intensity*-ICP [16], en el que se usa la intensidad de reflexión láser. El *Iterative Closest Line* (ICL) [17] es otra variante del ICP, cuya principal diferencia es que en vez de realizarse la combinación entre los puntos de las dos nubes, son líneas extraídas de los puntos de referencia los que se combinan con los puntos de la otra nube.

Todas estas técnicas de *scan matching* pueden ser utilizadas en espacios interiores [18], o exteriores [19]. Cabe destacar el trabajo realizado por Mallios *et al.* [20], en el que, basándose en técnicas probabilísticas de *scan matching*, proponen un método de localización y mapeo para un vehículo autónomo submarino utilizando *scans* proporcionados por un sónar.

Los avances en sensores impulsaron a los investigadores en robótica a enfrentarse al mapeo en 2D de grandes entornos, como el uso de sensores láser o de ultrasonidos utilizados por Bosse *et al.* [21] en su trabajo en grandes entornos cíclicos; así como aumentaron el interés por el desarrollo de mapas 3D, como por ejemplo, Nütcher *et al.* [19]. En su trabajo, desarrollaron un método para construir mapas a partir de escaneos 3D combinando el método de *scan matching* Iterative Closest Point (ICP) con un heurístico para detectar *bucles cerrados*[2] y un *método de relajación*[3] global. Otros métodos han sido desarrollados para construir mapas 3D. Algunos grupos de investigadores optaron por la sustitución de escáneres láser 3D por unos de 2D: Thrun *et al.* [22] se inclinaron por la combinación de varios escáneres láser 2D para generar representaciones volumétricas del entorno; mientras que Schadler *et al.* [23] emplearon un único escáner láser 2D en rotación continua.

Un factor importante a tener en cuenta en el mapeado, es la información visual. Varios investigadores han realizado trabajos teniendo en cuenta este tipo de información en diferentes ámbitos. May *et al.* [24] emplean cámaras *time-of-flight*[4]

[2] en inglés, *closed loop*. En robótica, se produce *loop closing* cuando el robot detecta que ya ha pasado por esa posición, tras haber realizado un recorrido "circular".

[3] en inglés, *relaxation method*. Son métodos iterativos utilizados para solucionar sistemas de ecuaciones.

[4] es un sistema que calcula la distancia entre la cámara y un objeto midiendo el "tiempo de vuelo" de una señal lumínica (por ejemplo, LEDs)

para construir mapas 3D del entorno. Un inconveniente de este tipo de cámaras es que presenta mucha carencia en precisión de medida. Henry *et al.* [25] utilizan una cámara RGB-D que genera imágenes RGB con información de profundidad. Proponen un algoritmo de optimización en el que se combinan características visuales y la alineación basada en la forma. Menegatti *et al.* [26] emplean un sistema de visión omnidireccional sobre un robot móvil, cuya finalidad es encontrar en un entorno determinado las distancias entre las transiciones de color más cercanas. Newman *et al.* [27] aprovechan las imágenes adquiridas a través de una cámara para detectar circuitos cerrados.

Uno de los sensores más utilizados para obtener información de profundidad es el *Microsoft Kinect.* Algunos trabajos se han centrado en realizar un análisis de dicho sensor, como por ejemplo Khoshelham *et al.* [28], quienes realizan un estudio sobre la resolución y exactitud de los datos de profundidad proporcionados por dicha cámara; mientras que en otros casos se ha empleado para mapear. Un ejemplo de este último uso puede ser el trabajo realizado por Newcombe *et al.* [29], en el que presentan un sistema de mapeado en tiempo real con condiciones variables de luminosidad del entorno.

Como ya se mencionó anteriormente, el problema de la optimización está presente a la hora de realizar mapas, y puede solucionarse empleando procesos iterativos. Pero aunque el uso de estos métodos está muy extendido, también se emplean técnicas heurísticas. Algunas de estas técnicas heurísticas son la optimización basada en la colonia de hormigas, los algoritmos genéticos o las estrategias evolutivas, entre otros. Con la optimización se busca al mejor elemento perteneciente a un conjunto, minimizando o maximizando una función. Este trabajo se centra en resolver el problema de la optimización mediante un algoritmo basado en estrategias evolutivas.

Capítulo 3.

RGB-D Scan Matching

3.1 El problema del *Scan Matching*

Para poder hacer una reconstrucción del entorno es necesario obtener información desde diferentes puntos de vista. La representación parcial obtenida desde estos puntos de vista se conoce como *scan* y, para poder construir una representación global del entorno, es necesario realizar una alineación de los diferentes *scans.*

El *scan matching* es un método que tiene como finalidad encontrar la transformación rígida (rotación y traslación) que alinea los puntos de los dos *scans.* En otras palabras, considere que el robot se encuentra en una cierta ubicación inicial Pos_{ref}, que será la de referencia, y en esa posición toma un escaneo Esc_{ref}, que también será el de referencia. Tras ejecutar la tarea, el robot se desplaza a una nueva posición, Pos_{actual}, en la que realiza un nuevo escaneo, Esc_{actual}. Con esta información, el problema del *matching* consiste en buscar una traslación T y una rotación ω, de manera que al aplicar la transformación sobre Esc_{actual}, este quede alineado con Esc_{ref}.

Los barridos proporcionados por los sensores tienen un elevado número de puntos, lo que implica un alto coste computacional. Con el fin de reducir este coste y de realizar mejor el *matching*, hay enfoques basados en puntos o en características previamente extraídas mediante los que se seleccionan los puntos para realizar la alineación. En función de si es necesario el tratamiento de la imagen previo a la extracción de puntos, podemos distinguir tres métodos. El primero se conoce como *point-based,* y es útil cuando no es necesario obtener información característica de los escaneos, ya que consiste en alinear directamente los puntos de cada nube. Por otro lado, si el entorno presenta rasgos geométricos destacables, sería necesaria la extracción de características previas al alineamiento. Este método se conoce como *featured-based.* El último método es una mezcla de los dos anteriores, en el que se combinan características y puntos de las nubes (*point to feature)*. De este modo, se puede seleccionar el método más adecuado en función del entorno; o de la utilidad del mapa.

Los métodos de *scan matching* se pueden dividir entre métodos locales, en los que se combinan escaneos tomados desde diferentes localizaciones; y globales,

en los que se utilizan el *scan* de la posición actual y un modelo global para realizar el *matching*. Es posible distinguirlos también en función de si el mapa es en dos dimensiones (2D) o tres (3D). Estos últimos pueden ser utilizados para trabajar en seis dimensiones (6D), tres de ellas se corresponden con las coordenadas lineales (x, y, z), y las otras tres con las angulares (ϕ , θ , ψ).

3.1.1 Algoritmos más comunes de *Scan Matching*

Iterative Closest Point (ICP)

Es una de las técnicas más utilizadas. Es un algoritmo iterativo, basado en la correspondencia entre puntos (*point-to-point based*). Se tienen dos nubes de puntos: la primera corresponde con el modelo global (M), y la segunda con los datos de la posición actual (S). Con estos datos, en cada iteración el algoritmo selecciona primero los puntos más cercanos como correspondencias, y posteriormente busca encontrar la transformación rígida (R_w, t) que minimiza la siguiente función de coste [30]:

$$E(R_w, t) = \sum_{i=1}^{N_M} \sum_{j=1}^{N_s} w_{i,j} \left\| m_i - \left(R_w * d_j + t\right) \right\|^2 \quad (3.1)$$

Donde:

$w_{i,j}$ determina la correspondencia entre dos puntos, de manera que $w_{i,j}$ tendrá valor 1 cuando el punto i del modelo global m corresponda con el punto j de los datos actuales d. Por el contrario, su valor será cero si no existe dicha correspondencia;

N_M es el número de puntos en el modelo global (M);

N_S es el número de puntos del *scan* actual (S).

Este algoritmo, recibirá como entradas dos barridos proporcionados por el sensor, el valor umbral del error y el criterio que detiene las iteraciones. Como resultado se obtendrá la transformación que alinea el modelo global con los datos. El primer paso del algoritmo es la inicialización de la transformación y de las dos nubes de puntos. Después se ejecuta el bucle principal, en el que en cada iteración se asocian los puntos de cada nube utilizando el criterio del vecino más cercano, seguidamente se calcula la transformación a través de la función de coste y se aplica la transformación calculada a los datos. Este bucle se realiza hasta que se cumple el criterio de fin.

Para facilitar la comprensión, en la *Figura 3* se muestra el pseudocódigo del algoritmo [31]:

Pseudocódigo algoritmo ICP

```
1  Inicializaicón de T^0
2  iteración = 1;
3  final = 0;
4  mientras final == 0 hacer
5      aplicar la transformación T^(iteración−1) al conjunto de datos S;
6      para todos d_j pertenecientes al conjunto S hacer
7          buscar el punto más cercano a d_j en el conjunto M;
8      fin
9      si iteración < valor máximo de iteraciones o error <
       valor del umbral entonces
10         final = 1;                    ■ fin del bucle principal
11     sino
12         calcular la transformación T^iteración que minimiza la función de coste;
13         iteración = iteración + 1;
14     fin
15 fin
16 devolver T^iteración
```

Figura 3. Pseudocódigo del ICP

Este algoritmo presenta algunas ventajas. Por un lado, no necesita la extracción previa de características ni un procesado previo de los datos. Por otro lado, debido a que su coste computacional no es muy elevado, puede utilizarse en sistemas de tiempo real. Un inconveniente es que el algoritmo es proclive a converger a un mínimo local, pudiéndose no encontrar la mejor solución.

Metric-based Scan Matching

Cuando se utiliza el ICP, es muy común el uso de la distancia Euclídea para establecer las correspondencias entre puntos. Un problema de este uso es que esta distancia no tiene en cuenta que aquellos puntos que se encuentran lejos del sensor, podrían encontrarse lejos de su correspondiente debido a las rotaciones del sensor. Minguez *et al.* [32] comprendieron que el principal problema del ICP residía en encontrar una manera de medir rotación y traslación para encontrar el vecino más cercano y aplicar la minimización, de manera que se capturase la traslación y rotación del sensor al mismo tiempo.

De este modo se define una nueva medida de la distancia en el espacio de la imagen del sensor, que tiene en cuenta rotación y traslación. La distancia entre dos

puntos se establece como la norma de la mínima transformación rígida que lleva de un punto a su correspondiente.

En el plano, una transformación rígida viene definida por un vector $q = (x, y, \theta), -\pi < \theta < \pi$, que representa la posición y la orientación del sensor en el plano. La norma de dicho vector viene dada por la siguiente ecuación: $\|q\| = \sqrt{x^2 + y^2 + L^2\theta^2}$ (3.2), donde L es un número real positivo.

La distancia entre dos puntos $p_1 = (p_{1x}, p_{1y})$ y $p_2 = (p_{2x}, p_{2y})$ es

$$d_p(p_1, p_2) = mín\{\|q\| \; tal \; que \; q(p_1) = p_2\} \qquad (3.3)$$

Donde

$$q(p_1) = \begin{pmatrix} x + \cos\theta * p_{1x} - \sin\theta * p_{1y} \\ y + \sin\theta * p_{1x} - \cos\theta * p_{1y} \end{pmatrix} \qquad (3.4)$$

Desarrollando, el conjunto de soluciones puede expresarse como:

$$\begin{cases} x = p_{2x} - p_{1x} - \theta * p_{1y} \\ y = p_{2y} - p_{1y} - \theta * p_{1x} \end{cases} \qquad (3.5)$$

Sustituyendo los valores de x e y en la ecuación (3.2), la distancia entre p_1 y p_2 resulta:

$$d_p(p_1, p_2) = \sqrt{\delta_x^2 + \delta_y^2 - \frac{(\delta_x * p_{1y} - \delta_y * p_{1x})^2}{p_{1y}^2 + p_{1x}^2 + L^2}} \qquad (3.6)$$

Esta distancia es utilizada también para aplicar el criterio de mínimos cuadrados.

Utilizando este método para realizar el *matching* entre dos escaneos 2D, se obtienen resultados más robustos y precisos, y se disminuye el tiempo de computación y convergencia.

Conditional Random Fields Matching (CRF-Matching)

CRF-Matching [6] es un método de *scan matching* basado en características (*featured-based*). Este planteamiento es capaz de considerar formas arbitrarias y rasgos característicos con el fin de realizar el *matching* entre los escaneos láser, convirtiendo las medidas individuales de un *scan* en nodos ocultos de un campo condicional aleatorio (conditional random field, CRF). La consistencia de la

asociación entre los dos *scans* se consigue a través de las conexiones entre nodos en el CRF.

Conditional random fields, son modelos gráficos desarrollados para clasificar datos, y modela la distribución condicional sobre las variables ocultas x dada una observación z, $p(x|z)$. CRF-Matching crea un campo condicional aleatorio, mostrado en la *Figura 4*, que contiene un nodo oculto x_i para cada punto del *scan*.

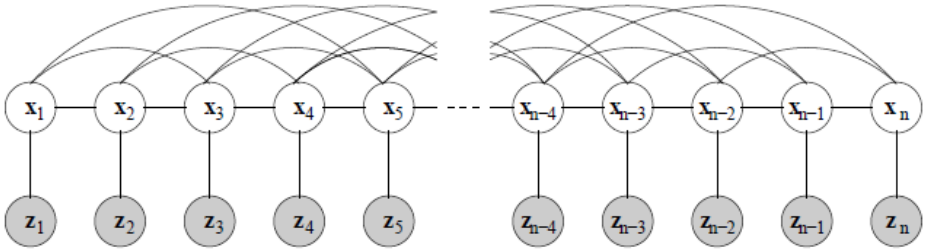

Figura 4. Representación gráfica de un CRF. Las observaciones, $\mathbf{z}_i$, se corresponden con la información obtenida de los dos escaneos. Los estados ocultos, $\mathbf{x}_i$, señalan las asociaciones entre puntos de los dos scans. [6]

Para calcular la transformación entre dos *scans* y estimar la posición, este método utiliza la incertidumbre sobre la asociación obtenida, incorporándola en el proceso de optimización por mínimos cuadrados. Siendo A y B los dos escaneos, la ecuación a minimizar queda:

$$E = \sum_{i=1}^{n} w_i \left(z_{A,i} * R_w + T - z_{B,a(i)} \right)^2 \qquad (3.7)$$

Donde

a(i) es el punto del *scan* B asociado con el punto *i*;

R_w y T son las matrices de rotación y traslación respectivamente;

w_i corresponde con la probabilidad de la asociación para el punto *i*.

Con este método, aquellos puntos que tienen mayor probabilidad de estar asociados correctamente estarán multiplicados por un coeficiente w_i mayor, y por tanto, tendrán más importancia para realizar la estimación de la posición.

Normal Distribution Transform (NDT)

Biber y Straßer [33] proponen un método para escaneos de dos dimensiones, que mide la probabilidad de que un punto exista en una determinada posición según una serie de distribuciones normales. Este método se conoce como *Normal Distribution Transform* (NDT). Al igual que en los métodos de celdas de ocupación, se busca subdividir el plano. Pero se diferencian en que mientras el NDT representa la probabilidad de medir una muestra para cada posición dentro de la célula, los algoritmos de ocupación reflejan la probabilidad de que una célula esté ocupada o no, por lo que el NDT proporciona una mayor precisión.

El algoritmo propuesto, *Normal Distribution Transform* (NDT), modela la distribución de todos los puntos reconstruidos de uno de los escaneos a partir de una serie de distribuciones normales locales. Inicialmente, el espacio del plano alrededor del robot se subdivide en celdas con un tamaño constante y, para cada celda, se realizan los siguientes pasos:

1 Reunir todos los puntos 2D ($x_{i=1\dots n}$) contenidos en la celda.
2 Calcular la media: $q = {}^{1}/_{n} * \sum_{i=0}^{n} x_i$. (3.8)
3 Calcular la matriz de covarianza:
$Covarianza = {}^{1}/_{n} \sum_{i=0}^{n} (x_i - q)(x_i - q)^t$. (3.9)

La probabilidad vendrá dada por la siguiente ecuación:

$$p(x) \sim exp\left(-\frac{(x-q)^t * \sum^{-1}(x-q)}{2}\right) \qquad (3.10)$$

Dados dos *scans*, para encontrar la transformación que los alinea, se realizan los siguientes pasos:

1 Se construye el NDT para el primer escaneo.
2 Se inicializa la estimación del valor de los parámetros.
3 Para cada muestra del segundo escaneo, se construye un mapa de los puntos 2D en el sistema del primer escaneo según los parámetros.
4 Se calculan las distribuciones normales para cada punto mapeado.
5 Se determina el valor de los parámetros evaluando las distribuciones, y sumando el resultado.
6 Se calcula un nuevo parámetro estimado intentado optimizar el resultado.
7 Se repiten los pasos 3 a 6 hasta la convergencia.

3.2 Utilidad del color en el Scan Matching

Para el desarrollo de este trabajo, se va a utilizar la solución implementada por Martín *et al.* [4] para la extracción de puntos característicos previos a la realización del *matching*. En este apartado se va a realizar una breve explicación del mismo, así como de las utilidades que presenta el uso del color en el *Scan Matching*. Si se desea ampliar la información sobre el método de extracción, se puede consultar dicho trabajo.

Existen varios procedimientos para seleccionar los puntos de interés en un barrido láser para realizar el *matching*. En los últimos años, gracias al desarrollo de sensores que pueden proporcionar información sobre el color y la profundidad a través de imágenes RGB-D. Como consecuencia de estos avances, se han desarrollado métodos que utilizan dicha información para seleccionar los puntos de interés.

En el espacio de color RGB y como se puede ver representado en la *Figura 5*, cada color es obtenido a partir de los colores primarios rojo (Red), verde (Green) y azul (Blue). Cada componente R, G y B toma un valor que oscila entre un máximo y un mínimo y, en función de estos valores, se obtendrá un color u otro. Así, por ejemplo, el color amarillo se obtiene con el valor mínimo de B, y el valor máximo de R y G ($Color\ (RGB) = (255, 255, 0)$). Un factor importante a tener en cuenta es que el espacio de color depende del dispositivo utilizado, es decir, un objeto no generará los mismos valores RGB en dos dispositivos diferentes.

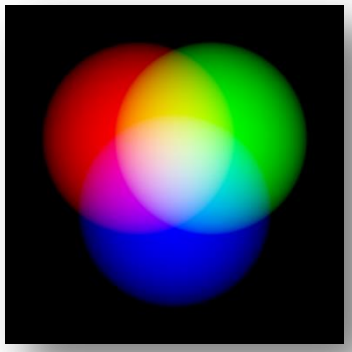

Figura 5. Espacio de color RGB [34]

Para poder conocer cuáles son los puntos más significativos de un *scan* se puede utilizar el espacio de color RGB para medir las variaciones de color entre un punto y su vecino más cercano. Esta diferencia de color viene dada por la siguiente ecuación:

$$d_c = \sqrt{\Delta R^2 + \Delta G^2 + \Delta B^2} \qquad (3.11)$$

Donde:

d_c es la diferencia de color;

$\Delta R, \Delta G\ y\ \Delta B$ son las diferencias de cada componente.

Si el valor de d_c es mayor que el valor del umbral, ese punto será clasificado como un punto significativo.

En este trabajo, se va a utilizar el *Delta E* divergence (ΔE) para medir las diferencias de color. El ΔE fue propuesto por la Comisión Internacional de la Iluminación, o CIE (*Commission Internationale de l'Éclariage*), y consiste en encontrar valores que representen una diferencia notable (*Just Noticeable Difference*, JND).

En 1931 el CIE propuso el CIE XYZ, un espacio de color basado en el RGB. Posteriormente, en 1976, fue propuesto el CIELAB (o CIE L*a*b*), mostrado en la *Figura 6* [35]. En este espacio de color, L* representa la luminosidad, cuyos valores están comprendidos entre 0 (negro) y 100 (blanco); a* las coordenadas rojo/verde, donde a positivo indica rojo y a negativo, verde; y b* las coordenadas amarillo/azul, en las que b positivo indica amarillo, y b negativo, azul. El CIELAB presenta la característica de ser más robusto ante cambios de iluminación que el espacio de color RGB.

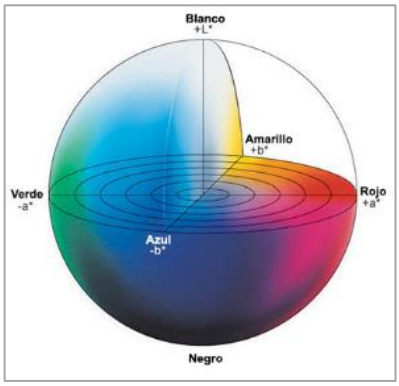

Figura 6. Espacio de color CIELAB [35]

A lo largo de los años se han ido desarrollando diferentes versiones del CIELAB: CIE76, CIE94, CIE2000 y CMC l:c (1984).

Como ya se ha mencionado, en este trabajo se va a utilizar la solución implementada por Martín *et al.* [4] del cálculo de ΔE para la selección de los puntos más significativos basada en CIE76. La ecuación de la diferencia de color entre dos puntos i, j viene dada por:

$$\Delta E^*_{i,j} = \sqrt{(L^*_2 - L^*_1)^2 + (a^*_2 - a^*_1)^2 + (b^*_2 - b^*_1)^2} \qquad (3.12)$$

De este modo, para cada punto del *scan* se calcula el valor de *Delta E* con respecto a su vecino más cercano en el espacio tridimensional. Si el valor resultante es mayor que el valor del umbral, este punto será clasificado como significativo, y se utilizará para realizar el *matching*; en el caso de que el valor sea menor, el punto será descartado.

Capítulo 4.

CMA-ES

4.1 Computación evolutiva

A principios de los años 30, algunos investigadores empezaron a interesarse por el proceso de la evolución de las especies, viéndolo como un proceso de aprendizaje, en el que la naturaleza dota a diferentes especies de diferentes medios, intentando hacerlas más aptas para la supervivencia. ¿Por qué unas sobreviven y otras no? Si partimos de la idea de la evolución, ¿por qué no desarrollar algoritmos que busquen solucionar problemas de optimización, basándose en el principio de la supervivencia del más apto? De este modo surgen los llamados Algoritmos Evolutivos (AE), y en la actualidad se emplean para solucionar un gran número de problemas, de diversos ámbitos.

La Computación Evolutiva (CE), campo que estudia los algoritmos evolutivos, es una de las ramas que, al igual que técnicas como las redes neuronales artificiales o la optimización de enjambre de partículas, forma parte de la Inteligencia Artificial y Computacional. Del mismo modo que la inteligencia artificial busca "imitar" la inteligencia natural, la CE trata de imitar la evolución biológica para resolver problemas de optimización.

Esta técnica se basa en el paradigma del neodarwinismo [36], el cual establece que la historia de muchas de las especies que conforman nuestro planeta puede explicarse a través de una serie de procesos estadísticos que actúan sobre las poblaciones y las especies. Estos procesos son la reproducción, propiedad inherente de cualquier forma de vida; la mutación, debida a la continua combinación de genes a través de la reproducción; la competencia, producida por la limitación de espacio disponible; y finalmente la selección de los individuos mejor adaptados. En la *Tabla 1* se muestran las analogías entre naturaleza y los AE.

Naturaleza	Algoritmos evolutivos
Individuo	Solución al problema
Población	Conjunto de soluciones
Adecuación o *fitness*	Calidad de la solución
Cromosoma	Representación de una solución
Gen	Componente de la representación de la solución
Mutación y recombinación	Operadores de búsqueda
Selección natural	Guardar y mantener la mejor solución

Tabla 1. Relación entre los algoritmos evolutivos y la evolución natural [37]

Todos los algoritmos evolutivos siguen el mismo esquema, también mostrado en la *Figura 7*:

1. Inicialización: se genera una población inicial, de manera aleatoria o con datos iniciales conocidos.
2. Evaluación: los datos de la población inicial son evaluados y clasificados.
3. Bucle principal: esta parte del algoritmo se repite hasta que se encuentra la solución óptima (o de la calidad que requiera el problema estudiado) o cuando se han alcanzado un número determinado de iteraciones (generaciones).

Figura 7. Diagrama de flujo de un algoritmo evolutivo [37]

- Selección de progenitores: una vez han sido clasificados los datos, se seleccionan aquellos que son más aptos para ser cruzados
- Recombinación: equivale a la reproducción de la población, mezclando los genes y aumentando las posibilidades de que surjan nuevos individuos o poblaciones óptimas.
- Mutación: se producen diferentes variaciones en las poblaciones.
- Evaluación de la función de *fitness*: con ella se evalúa cómo de bueno o malo es un individuo
- Selección de los supervivientes y reemplazo: se seleccionan los individuos que son más aptos para la supervivencia tras haber sufrido la mutación y evaluación.

Existen principalmente tres tipos algoritmos evolutivos: la Programación Evolutiva, los Algoritmos Genéticos y las Estrategias Evolutivas.

4.1.1 Programación evolutiva

La Programación Evolutiva (PE) fue introducida en 1966 por L.J Fogel, A.J. Owens y M.J. Walsh. En este método, las posibles soluciones se representaban como máquinas de estado finito sencillas, y avanzaba mutando aleatoriamente una de las máquinas simuladas y conservando la mejor. La evolución de dichos autómatas se realizaba teniendo en cuenta el "entorno" al que estaban expuestos.

Los pasos que sigue el algoritmo son los siguientes:

1 Generar de manera aleatoria la población inicial.
2 Aplicar la mutación.
3 Calcular la aptitud de cada "hijo".
4 Seleccionar los elementos más aptos para conservarlos.

Como se puede ver, en los pasos anteriores no existe la "recombinación". Esto es debido a que la PE es una conceptualización de la evolución, pero realizada a nivel de especies, partiendo de la premisa de que en la naturaleza no se pueden cruzar dos especies diferentes.

En la PE, la inteligencia es entendida con un comportamiento adaptativo, y se le da más importancia a los "vínculos" entre padres e hijos, más que a conseguir recrear operadores genéticos específicos. En la *Figura 8* se muestra el pseudocódigo del algoritmo.

Algoritmo 5 Algoritmo básico de la Programación Evolutiva

$t = 0$
generar una población inicial $\mathbf{X}_t$
mientras CFin = *falso* **hacer**
  mutar $\mathbf{X}_t$
  evaluar $\mathbf{X}_t$
  seleccionar elementos de $\mathbf{X}_t$
  $t = t + 1$
fin mientras

Figura 8. Pseudocódigo de un algoritmo de programación evolutiva [38]

Algunas de las aplicaciones de este método son problemas de predicción, reconocimiento de patrones y control automático [36,38].

4.1.2 Algoritmos genéticos

Los Algoritmos Genéticos (AG) es otra de las técnicas que conforman la Computación Evolutiva. Fueron introducidos por John Holland en 1975, con la publicación de "Adaptación en Sistemas Naturales y Artificiales". Son métodos adaptativos utilizados para solucionar problemas de optimización, en los que los "genes" tienen un papel muy importante en la supervivencia [39].

La naturaleza se rige por la ley de la supervivencia del más fuerte. Los diferentes individuos de una población compiten por recursos como el alimento, así como dentro de una especie compiten por encontrar una pareja. De este modo, aquellos individuos que tienen una mayor predisposición genética a sobrevivir y a

encontrar pareja tendrán una mayor descendencia. Los genes de los más aptos se mantendrán en generaciones futuras, surgiendo nuevos individuos con mejores características. Aquellos que están menos adaptados, tienen menos probabilidad de transmitir su material genético.

Los AG buscan emular este comportamiento. Se trabaja con poblaciones en las que cada individuo es una posible solución del problema. Cada uno de estos individuos es evaluado a través de una función de *fitness*, y se le asigna un valor según su aptitud. Aquellos que están más cualificados serán utilizados para la "reproducción" con el fin de que la nueva población sea mejor que la anterior y poder encontrar la solución óptima [40]. En la *Figura 9* se muestra el pseudocódigo de un AG básico:

Pseudocódigo Algoritmo Genético

```
1.  Generar una población inicial;
2.  Evaluar la aptitud de cada individuo;
3.  final = 0;
4.  mientras final == 0 hacer
5.      para el tamaño de la población hacer
6.          seleccionar los dos mejores individuos de la población;
7.          realizar el cruce obteniendo dos descendientes;
8.          mutar los descendientes;
9.          evaluar los descendientes mutados;
10.         genera nueva población;
11.     fin
12.     si la población a convergido entonces
13.         final = 1;
14.     fin
15. fin
```

Figura 9. Pseudocódigo AG

4.1.3 Estrategias Evolutivas

El último de los métodos son las Estrategias Evolutivas (*Evolution Strategy* o *Evolutionary Strategies*, ES), desarrolladas por Rechenberg en 1965 [41] con el fin de resolver problemas hidrodinámicos complejos, y planteados como una heurística de optimización que se basa en la idea de adaptación y evolución. En este primer algoritmo planteado, conocido como (1+1)-ES, se empleaba un único padre y un único hijo. El descendiente competía con el progenitor, y el mejor de los dos era el que se mantenía [38]. En la *Figura 10* se muestra el pseudocódigo de este algoritmo.

Algoritmo 6 Algoritmo $(1+1)-$EE

Requiere: k, m
  $t = 0$
  $p_s = 0$
  inicializa $\mathbf{x}_t = (x_1, \ldots, x_n)$ aleatoriamente
  evalua $f(\mathbf{x}_t)$
  mientras $t \leq m$ **hacer**
    mutar $\mathbf{x}_{t-mut} = \mathbf{x}_t + \mathbf{N}(0, \Sigma_t)$
    evaluar $\mathbf{x}_{t-mut}$
    $\mathbf{x}_t =$ **mejor**$(\mathbf{x}_{t-mut}, \mathbf{x}_t)$
    si $\mathbf{x}_t = \mathbf{x}_{t-mut}$ **entonces**
      $p_s = p_s + 1$
    fin si
    $t = t + 1$
    si t **mod** $k = 0$ **entonces**

$$\Sigma_t = \begin{cases} \Sigma_t / c & \text{si } p_s/k < 1/5 \\ c\Sigma_t & \text{si } p_s/k > 1/5 \\ \Sigma_t & \text{si } p_s/k = 1/5 \end{cases}$$

      $p_s = 0$
    si no
      $\Sigma_t = \Sigma_{t-1}$
    fin si
  fin mientras

Figura 10. Pseudocódigo del algoritmo (1+1)-ES [38]

En este algoritmo, a cada elemento de x_t se le aplica una mutación aleatoria con una distribución normal. En cuanto a la mutación de las varianzas, se deben transformar de manera que si se está lejos de la solución, se obtengan valores grandes; y pequeños si se está próximo a la solución. En el momento en el que se empiezan a generar malas soluciones, implica que la solución que se había obtenido como mejor es difícil de mejorar [42].

A partir de este, se propusieron dos métodos:

- Técnicas (μ, λ): se escogen los μ mejores elementos del conjunto de λ individuos.
- Técnicas $(\mu + \lambda)$: se escogen los μ mejores del conjunto formado por descendientes λ y los progenitores μ.

Mientras que en los AG se potencia la búsqueda de la solución a partir del cruce de genes, en las ES se potencia a través de las mutaciones. Se parte de una población P formada por un conjunto de individuos. Cada individuo está compuesto por una solución o vector del parámetro del objeto (rasgos visibles) y parámetros internos (rasgos no visibles), y un valor de la función de coste. En algunos casos, la población está formada por un solo individuo. Estos individuos se conocen como *padres* o *descendientes* en función de en qué paso del algoritmo se esté. El procedimiento general es el siguiente:

1. Uno o varios padres son seleccionados de la población y se generan los descendientes por duplicación y recombinación de estos padres.

2. Los nuevos descendientes mutan y pasan a formar parte de la población.
3. Una selección reduce la población al tamaño original [43].

4.2 Covariance Matrix Adaptation Evolution Strategy

4.2.1 Introducción

En estadística, se dice que dos variables están relacionadas si varían de manera conjunta. La covarianza es la medida que indica cuánto han cambiado dos variables aleatorias conjuntamente. Sean $X = (x_1, x_2, \dots, x_n)$ e $Y = (y_1, y_2, \dots, y_n)$ dos variables aleatorias, la ecuación de la covarianza quedaría definida como:

$$covarianza_{x,y} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x}) * (y_i - \overline{y}) \qquad (4.1)$$

Donde:

$\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$ (4.2) es la media de la variable *X*,

$\overline{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ (4.3) es la media de la variable *Y*,

n es el número de elementos de *X* e *Y*.

Si se desarrolla la *Ecuación 4.1*, se obtiene una forma más sencilla para el cálculo de la covarianza:

$$covarianza_{x,y} = \frac{1}{n}\left(\sum_{i=1}^{n} x_i y_i \; - \overline{x}\sum_{i=1}^{n} y_i - \overline{y}\sum_{i=1}^{n} x_i + n\overline{xy}\right)$$

$$= \frac{1}{n}\left(\sum_{i=1}^{n} x_i y_i - n\overline{xy}\right) \quad (4.4)$$

Analizando las expresiones, se puede determinar que:

- Si los valores de *X* e *Y* tienden a crecer en la misma dirección, el signo de la covarianza será positivo.
- Si los valores de *X* tienden a disminuir con el crecimiento de los valores de *Y* o viceversa, el signo será negativo.
- Si no hay relación clara entre el crecimiento y decrecimiento de las variables, por ejemplo, *X* crece pero no se aprecia variación en *Y*, la covarianza estará próxima a cero [44].

Una población está compuesta por un conjunto de elementos como los X e Y descritos anteriormente. Si en cada elemento de dicha población se miden diferentes variables, se dice que se tiene una población multivariante.

Sea X una matriz de datos, en la que cada fila n es un elemento de la población, y cada columna m una variable observable:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix} \quad (4.5)$$

La matriz de covarianza establece las relaciones que existen entre los elementos de la matriz X. Es una matriz simétrica de orden m, en la que los elementos de la diagonal representan las varianzas de las variables, y los elementos restantes corresponden a los valores de las covarianzas entre todos los pares de variables. La matriz de covarianza C es de la forma:

$$C = \begin{bmatrix} c_1^2 & c_{12} & \cdots & c_{1m} \\ c_{21} & c_2^2 & \cdots & c_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m1} & c_{m2} & \ldots & c_m^2 \end{bmatrix} \quad (4.6)$$

Y se calcula utilizando la siguiente ecuación [45]:

$$C = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x}) * (x_i - \overline{x})^t \quad (4.7)$$

4.2.2 Descripción del método

El *Covariance Matrix Adaptation-Evolution Strategy* (CMA-ES) [46, 47, 48, 49] es un algoritmo evolutivo de optimización de funciones no lineales y no convexas en dominio continuo. Inicialmente pensado para tamaños pequeños de población, este algoritmo genera individuos mediante una distribución normal multidimensional (*Ecuación 4.8*). A partir de esta distribución, se forma una población de λ individuos y, utilizando la función de *fitness* o *cost* $f(x)$: $\mathcal{R}^n \rightarrow \mathcal{R}$, se escogen los μ mejores. Con estos individuos, el algoritmo trabajará para generar la siguiente población.

$$x_k^{(g+1)} \sim m^{(g)} + \sigma^{(g)}\mathcal{N}\left(0, C^{(g)}\right) \sim \mathcal{N}\left(m^{(g)}, \left(\sigma^{(g)}\right)^2 C^{(g)}\right) \quad (4.8)$$

$$\text{para } k = 1,2,3 \ldots \lambda.$$

Donde:

$g=0,1,2\ldots$ es el número de la generación;

$x_k^{(g+1)}$ denota el descendiente k de la generación (g+1);

$m^{(g)}$ corresponde con la media de la generación g;

$\sigma^{(g)}$ es la desviación estándar de la generación g;

$C^{(g)}$, matriz de covarianza nxn de la generación g;

$\sim$ indica que a ambos lados se encuentra la misma distribución.

El algoritmo calcula dos parámetros principales, para poder generar una distribución normal:

- El primer parámetro es la media de la distribución. Esta se obtiene como una media ponderada de los μ mejores individuos, de manera que a mejor valor de función de coste, mayor valor de ponderación se le da a dicho individuo. La expresión empleada para su cálculo es la siguiente:

$$m^{(g+1)} = \sum_{i=1}^{\mu} w_i x_{i:\lambda}^{(g+1)} \qquad (4.9)$$

Donde:

w_i es el valor del coeficiente de ponderación del elemento i calculado como $\sum_{i=1}^{\mu} w_i = 1, \qquad w_1 \geq w_2 \geq \cdots \geq w_\mu \geq 0;$ (4.10)

$x_{i:\lambda}^{(g+1)}$ denota el *i-ésimo* mejor elemento perteneciente al conjunto $x_1^{(g+1)} \ldots x_\lambda^{(g+1)}$, de manera que, siendo f la función de coste, $f\left(x_{1:\lambda}^{(g+1)}\right) \leq f\left(x_{2:\lambda}^{(g+1)}\right) \leq \cdots \leq f\left(x_{\lambda:\lambda}^{(g+1)}\right)$;

$m^{(g+1)}$ representa la media de la generación (g+1).

- El otro, es la matriz de covarianzas. Para el cálculo se emplean, también, los μ mejores elementos de la población. La media utilizada es la de la población anterior, ya que se pretende que estime las varianzas de los pasos muestreados.

$$C_\mu^{(g+1)} = \frac{1}{\mu}\sum_{i=1}^{\mu} w_i \left(x_{i:\lambda}^{(g+1)} - m^{(g)}\right)\left(x_{i:\lambda}^{(g+1)} - m^{(g)}\right)^t \qquad (4.11)$$

En el algoritmo, mientras que la media indica la localización actual de la búsqueda y debe ir mejorando según se avance hacia la solución, la covarianza se emplea para encaminar dicha búsqueda y controlar las mutaciones.

En la *Figura 11* se muestra un ejemplo de una distribución de $\lambda = 150$ elementos, de los cuales se escogen los $\mu = 50$ mejores y una ponderación realizada de acuerdo a $w_i = \frac{1}{\mu}$. La estimación de la matriz de covarianza se realiza minimizando la función $\mathrm{f(x)} = -\sum_{\mathrm{i=1}}^{2} \mathrm{x_i}$ (4.12).

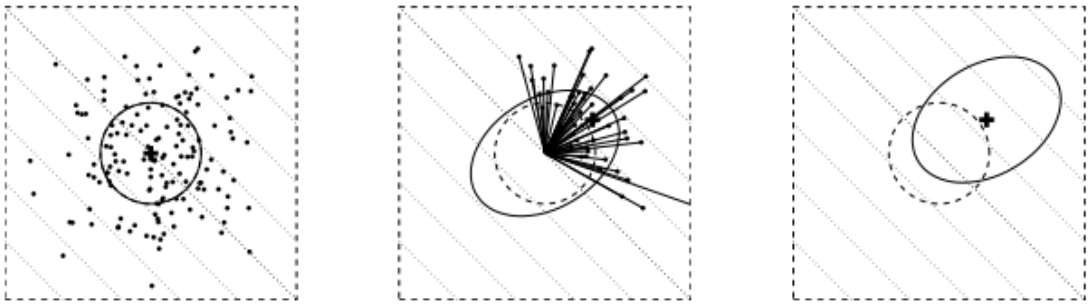

Figura 11. Estimación de la matriz de covarianza de $\boldsymbol{f}(\boldsymbol{x}) = -\sum_{i=1}^{2} \boldsymbol{x}_i$. A la derecha, muestra de $\boldsymbol{\lambda = 150}$ elementos. En el medio, μ=50 elementos seleccionados para el cálculo. A la izquierda, distribución de la siguiente generación. [49]

Hansen *et al.* desarrollaron algunas mejoras a la hora de obtener la matriz. La primera de estas mejoras se conoce como *Rank-μ-Update*. Para este caso, pensaron que no se podía obtener una buena aproximación de la matriz teniendo en cuenta solo la información "del momento". Para solucionarlo, se introdujo el uso de la información de la covarianza de generaciones previas, de manera que las más recientes tuvieran más peso. La siguiente se denomina *Rank-One-Update*, en la que se introduce la idea de "paso" (*step*) entre generaciones. A diferencia del caso anterior, solo se utiliza la información del paso anterior.

La finalidad de los dos planteamientos anteriores es poder reducir el tamaño de la población necesaria para el cálculo de la matriz, de manera que el resultado obtenido sea fiable para el cálculo de las generaciones posteriores. La tercera opción de mejora consiste en controlar el tamaño de los pasos (*Step-Size Control*), con el fin de disminuir el número de generaciones necesarias para llegar a una buena solución, sin caer en mínimos locales. Para controlar el tamaño de los pasos, se compara con el tamaño que tendría un camino aleatorio. Si es más corto, los pasos se anulan entre ellos y se debe disminuir el tamaño del paso. Si en cambio es más largo, los pasos apuntan a direcciones similares, es decir, existe correlación entre ellas y se podría reducir el número de pasos aumentando el tamaño de los mismos.

En la *Figura 12* se muestra un ejemplo de tres caminos de seis pasos, en los que, para llegar al punto final, se han ido escogiendo diferentes direcciones en cada paso. Se puede ver la diferencia de la longitud de cada una de las opciones.

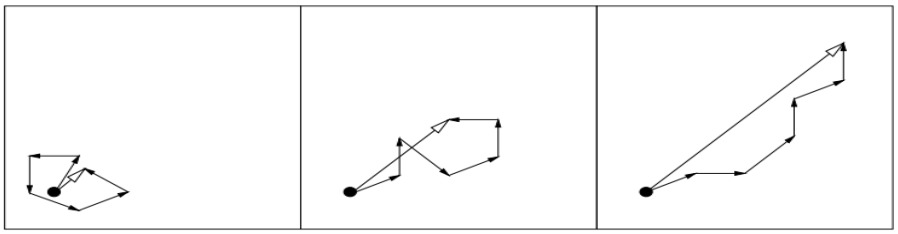

Figura 12. Ejemplo de un camino de seis pasos [49]

En la *Figura 13* se puede ser el pseudocódigo del algoritmo *Covariance Matrix Adaptation-Evolution Strategy*, CMA-ES. Seguido de la imagen se detallan los pasos seguidos por el algoritmo.

Pseudocódigo CMA-ES

1. *Generar una población inicial aleatoria;*
2. *Cálculo de la media y la matriz de covarianza de la población inicial;*
3. $iteración = 1;$
4. ***mientras*** $iteración < IteraciónMax$ ***hacer***
5. *Cálculo de la distribución normal* $\mathcal{N}(m^g, C^g);$
6. ***para*** *cada elemento de la población* ***hacer***
7. *Cálculo de la función de coste;*
8. ***fin***
9. *Ordenar los elementos de acuerdo al valor del coste;*
10. ***para*** *los* μ *mejores elementos* ***hacer***
11. *Cálculo de la media* $m^{(g+1)}$ *de acuerdo a la Ec.* 4.9;
12. ***fin***
13. ***para*** *los* μ *mejores elementos* ***hacer***
14. *Cálculo de la matriz de covarianza* $C^{(g+1)}$ *de acuerdo a la Ec.* 4.11;
15. ***fin***
16. $m^{(g)} = m^{(g+1)};$
17. $C^{(g)} = C^{(g+1)};$
18. ***devolver*** *el valor de la población;*
19. $iteración = iteración + 1;$
20. ***fin***

Figura 13. Pseudocódigo del CMA-ES

El algoritmo funciona de la siguiente manera. Inicialmente se genera una población inicial aleatoria. Para poder hacer el cálculo de la media y la matriz de covarianza iniciales, es necesario calcular el coste de cada uno de los elementos de la población inicial. Una vez calculado, los elementos dicha población son

ordenados, teniendo más importancia aquellos que tienen un menor valor de función de coste. Es decir, se ordenan de menor a mayor según el coste.

Después de ser ordenados, se escogen los μ mejores elementos para el cálculo de la media inicial con la *Ecuación 4.9*. La matriz de covarianza inicial se calcula empleando la *Ecuación 4.11*, con la diferencia de que, al no tener población anterior, se utiliza la media de la población inicial.

A partir de ahí, comienza el bucle principal del algoritmo. El primer paso es calcular la nueva población a partir de una distribución normal. Si el algoritmo se encuentra en la primera iteración, se calcula utilizando los valores iniciales de la media y de la matriz de covarianza; sino, se utiliza el valor calculado en la iteración anterior.

A continuación, se obtiene el valor de coste de cada elemento, y se ordenan como en el paso inicial, con el fin de volver a escoger los μ mejores individuos de la nueva población. Con estos elementos, se realiza el cálculo de la media y la matriz de covarianza de la nueva generación, $m^{(g+1)}$ y $C^{(g+1)}$ respectivamente. En este caso, al existir una población anterior, para obtener $C^{(g+1)}$ se utiliza la media de la generación anterior, $m^{(g)}$. Este bucle principal se repite hasta que se alcanza el número máximo de iteraciones.

Capítulo 5.

CMA-ES Scan Matching

5.1 Introducción

En este capítulo se va a realizar una descripción del proceso seguido para la implementación del algoritmo *CMA-ES Scan Matching* utilizando las propiedades del color.

El método del *Scan Matching* busca minimizar el error de localización de un robot en un entorno determinado, a través de la alineación de los puntos pertenecientes a dos imágenes de dicho entorno. Con esta alineación se pretende encontrar la transformación rígida (traslación y rotación) que lleva de la posición del primer *scan* a la posición del segundo, y así estimar la posición.

Como ya se mencionó en capítulos anteriores, los métodos de *scan matching* pueden ser globales o locales. Para la realización de este trabajo se utiliza un método global, es decir, se busca la correspondencia entre puntos de un barrido y el modelo global, minimizando la distancia entre dichos puntos.

El proceso emplea dos barridos 3D de una base de datos que contiene imágenes proporcionadas por el sensor *Kinect*. Uno de ellos se corresponde con el modelo, y el otro con el *scan* tomado desde una cierta posición. Antes de empezar a buscar la transformación, el algoritmo realiza un proceso de selección de los puntos característicos, midiendo la diferencia de color entre puntos (detallado en el *Capítulo 3*). Una vez seleccionados, la correspondencia entre dichos puntos se busca utilizando un algoritmo evolutivo, mediante el cual, utilizando una función de coste, se evalúa de manera iterativa si la aproximación de la transformación es próxima a la solución óptima o no.

Al algoritmo evolutivo empleado para este trabajo, el *Covariance Matrix Adaptation-Evolution Strategy*, ya ha sido explicado en el *Capítulo 4* de manera general. En este capítulo se va a realizar una descripción de la solución implementada del algoritmo. Para facilitar la compresión, en la *Figura 14* se muestra el pseudocódigo del algoritmo implementado.

5.2 Implementación del algoritmo

Se parte de una población de N_p individuos, en la que cada individuo representa una posible solución. Dado que el robot se encuentra en un entorno en tres dimensiones, su posición tiene 6 grados de libertad (DoF). De este modo, cada punto estará definido por 6 componentes: 3 corresponden a la posición x, y, z en coordenadas cartesianas, y los otros tres a la orientación proporcionada por los ángulos de Euler ϕ, θ, ψ.

$$pob_i^k = \left(x_i^k, y_i^k, z_i^k, \phi_i^k, \theta_i^k, \psi_i^k\right) \qquad (5.1)$$

Donde pob_i^k representa el elemento i de la iteración k.

La población inicial se situará en una esfera cercana a la posición estimada por los sensores internos del robot, y se calculará de manera aleatoria. El objetivo es corregir el error de posición de estos sensores. La generación de la población se inicia en la *línea 2* del algoritmo. Dicha población tendrá la forma de una matriz de N_p filas y 7 columnas: la primera, se corresponde con el valor del coste; las tres siguientes corresponden a las coordenadas x, y, z; y las tres restantes, a los ángulos de Euler φ, ϕ, ψ.

$$\begin{bmatrix} e_1 & x_1 & y_1 & z_1 & \phi_1 & \theta_1 & \psi_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ e_{N_p} & x_{N_p} & y_{N_p} & z_{N_p} & \phi_{N_p} & \theta_{N_p} & \psi_{N_p} \end{bmatrix} \qquad (5.2)$$

Para buscar las correspondencias entre puntos de ambos barridos, se va a emplear la transición de colores, es decir, se buscan puntos que presentan las mismas características de color (*línea 1* del algoritmo). Previamente a la obtención del coste, es necesario corregir los datos. Para ello, se le aplica una función, *data_correction_by_*pose implementada por Martín *et al.* [4], mediante la que se trasladan y rotan los puntos del barrido, hasta alinearlos con los de la población pob. La función de coste es aplicada a todos los elementos de la población, y calcula su valor midiendo las distancias entre un punto y su correspondiente (*línea 4*).

Una vez se ha obtenido el valor de la función de coste, es necesario ordenar los individuos (*línea 6*) de la población de menor a mayor con el fin de escoger los mejores elementos para calcular la media y la matriz de covarianza iniciales. Como se puede ver en la *línea 15*, el valor inicial de la matriz se realiza utilizando la media de la misma generación, ya que el valor de la variable $media\_ant = 0$ y era necesario aproximarlo a algún valor.

A partir de ahí, comienza el bucle principal del algoritmo (*línea 17*), que realizará hasta un determinado número máximo de iteraciones. El primer paso del bucle es el cálculo de la distribución normal, utilizando los valores de las variables $media\_ant$ y cov_ant (*línea 18*). A partir de esta distribución se obtienen los valores de las coordenadas cartesianas y los ángulos de Euler de cada punto.

$$pob^k \leftarrow normal\_distrib(media\_ant\ , cov\_ant) \quad (5.3)$$

La primera columna de la matriz (5.2) anterior, se completa aplicando la función de coste. La función de coste a minimizar es la derivada de la distancia entre los puntos de correspondencia característicos seleccionados a partir de las transiciones de color.

$$e = \sum_{c=1}^{C} d\left(m_{i,j}, d_{i,j}\right)^2 \qquad (5.4)$$

Donde:

e es el valor del coste;

C representa el número de correspondencias;

$d\left(m_{i,j}, d_{i,j}\right)$ denota la distancia entre dos puntos.

Una vez se tiene el valor del coste, los datos se ordenan de manera que aquellos individuos de la población que tienen un menor coste ocupen las primeras posiciones de la matriz (*línea 22*). Con el fin de ir tendiendo siempre a la mejor solución, se ha implementado una condición de if en la que se compara el mejor elemento de la población de la iteración k con el mejor elemento de la mejor población obtenida hasta el momento (*líneas 23 a 29*). En el caso de estar en la primera iteración, la matriz de la mejor población se inicializa al valor de pob_orden^1.

Con los mejores $N_p/4$ elementos de la matriz $best\_pob$ se calcula la media de la población, y la matriz de covarianza.

$$media\_sig = media\_sig + w_i * best\_pob_i^k \qquad (5.5)$$

$$cov\_sig = cov\_sig + w_i * \left[\left(best\_pob_i^k - media\_ant\right)^t \left(best\_pob_i^k - media\_ant\right)\right] \quad (5.6)$$

Antes de llegar al final de bucle principal, los valores de la media y la matriz de covarianza se actualizan, ya que para obtener la distribución normal se emplean

los valores calculados de la generación anterior (*líneas 34 y 35*). Para que el bucle *while* avance, es necesario incrementar la variable k.

- **Cálculo de los pesos**

Para realizar el cálculo de la media y de la matriz de covarianza según el método, es necesario dar una ponderación a los mejores elementos de la población, de manera que los que obtienen un menor valor de coste, se les da una ponderación mayor.

Según el método [49], el sumatorio de los pesos debe ser unitario: de modo que se cumpla que $w_1 \geq w_2 \geq \cdots \geq w_\mu \geq 0$. Para las pruebas realizadas con el algoritmo, se ha empleado la siguiente ecuación:

$$w_i = \sum_{n=1}^{\mu} \alpha * (1-\alpha)^{n-1} \quad (5.7)$$

Donde:

α es el valor del coeficiente de ponderación;

μ es el número de mejores elementos escogidos.

Pseudocódigo CMA-ES Scan Matching utilizando las propiedades del color

$modelo\_tc, datos\_tc \leftarrow extraer\_TransicionesColor(modelo, datos)$
▫Extracción de las transiciones de color
for $i = 1{:}N_p$ ***do***
  $pob_i^1 \leftarrow inic\_pob(datos_{pos_{inicial}})$ ▫Inicialización de la población
  $e_i^0 \leftarrow coste(modelo\_tc, datos\_tc, pob_i^1)$
end for
$pob\_orden^1 \leftarrow odenar(e^0)$
$a = valor\_peso;$
for $a = 1{:}10$ ***do***
  $w_a \leftarrow calculo\_pesos$ ▫Se calculan las ponderaciones
end for
▫Cálculo media inicial
for $i = 1{:}10$ ***do***
  $media\_ant = media\_ant + w_i * pob\_orden_i^1$ ▫Media de los 10 mejores
end for
▫Cálculo de la matriz de covarianza inicial
for $i = 1{:}10$ ***do***
  $cov\_ant = cov\_ant + w_i * [(pob\_orden_i^1 - media\_ant)^t (pob\_orden_i^1 - media\_ant)]$
  ▫Covarianza de los 10 mejores
end for
while $(k < iterMax)$ ***do***
  $pob^k \leftarrow normal\_distrib(media\_ant, cov\_ant)$
  ▫Cálculo de la distribución normal
  for $i = 1{:}N_p$ ***do***
    $e_i^k \leftarrow coste(modelo\_tc, datos\_tc, pob_i^k)$ ▫Cálculo coste
  end for
  $pob\_orden^k \leftarrow odenar(e^k)$ ▫Ordenar de menor a mayor
  ▫Selección de la mejor población
  if $k == 1$ ***then***
    $best\_pob^1 = pob\_orden^1$ ▫Inicialización mejor población
  else
    if $pob_{orden_{1,1}}^k \leq best_{pob_{1,1}}$ ***then***
      $best\_pob^k = pob\_orden^k$
    end if
  end if
  ▫Cálculo de la media y la covarianza
  for $i = 1{:}10$ ***do***
    $media\_sig = media\_sig + w_i * best\_pob_i^k$ ▫Media de los 10 mejores
    $cov\_sig = cov\_sig + w_i * \left[(best\_pob_i^k - media\_ant)^t (best\_pob_i^k - media\_ant)\right]$
    ▫Covarianza de los 10 mejores
  end for
  $media\_ant = media\_sig$
  $cov\_ant = cov\_sig$
  $bestmen \leftarrow best\_pob^k$
  $k = k + 1$
end while

Figura 14. Pseudocódigo CMA-ES Scan Matching utilizando las propiedades del color

Capítulo 6.

Resultados experimentales

Para realizar los experimentos, el algoritmo se ha implementado empleando la herramienta MATLAB. Los datos que necesita el algoritmo para funcionar se obtienen de una base de datos que contiene 2900 barridos 3D de un entorno real. Estos *scans* han sido obtenidos con un sensor *Kinect* [4].

En este capítulo, se van a exponer las diferentes soluciones implementadas, así como sus resultados. Primero, se deben definir los parámetros del algoritmo.

- **Parámetros de procesamiento de imágenes**

En este trabajo, se ha empleado un método de procesamiento previo de imágenes desarrollado por Martín *et al.* [4], con el fin de seleccionar los puntos característicos de los barridos, y disminuir así el coste computacional del algoritmo.

Esta selección se realiza midiendo las transiciones de color entre un punto, y sus vecinos más cercanos, de manera que, cuando una de estas diferencias es mayor que el valor del umbral, ese punto se clasifica como una "transición de color" y se define como punto característico. Aquellos que presenten un valor menor que el umbral, serán descartados.

El valor de umbral definido por Martín *et al.* [4] de la variable *Delta E* es $\Delta E^*_{ab} = 10$, y un radio de muestreo de 0.1.

- **Tamaño de la población**

Para elegir el tamaño de la población, se realizan varios experimentos con un número máximo de iteraciones de 50. Estas primeras pruebas se realizan con los *scans* 50-60 y utilizando el algoritmo básico, calculando en la primera iteración la media y la matriz de covarianza de la población total sin ponderar.

Tamaño de población	Mejor coste
10	1.7689
30	0.811871
50	0.57187
75	0.53546
100	0.457484

Tabla 2. Coste en función del tamaño de la población

En la *Tabla 2* se muestran los resultados obtenidos con dichas pruebas. El tamaño de la población elegido será de 100 individuos.

- **Número de iteraciones**

El número de iteraciones debía ser lo suficientemente alto como para que el algoritmo pueda encontrar la mejor solución. Ya se ha comprobado para el tamaño de la población que con un número máximo de 50 iteraciones, se podían obtener unos resultados de coste que comenzaban a ser buenos. Por este motivo, se decidió doblar el número de iteraciones, con el fin de obtener una mejor solución.

- **Valor del parámetro *alfa*, α**

Como ya se explicó en el capítulo anterior, la fórmula por la que se obtienen los valores de las ponderaciones es la siguiente:

$$w_i = \sum_{n=1}^{\mu} \alpha * (1-\alpha)^{n-1} \quad (5.7)$$

Para poder utilizarla, es necesario dar un valor al parámetro *alfa*, valor de la ponderación del mejor elemento de cada población.

Para este trabajo, diferentes pruebas fueron realizadas con el fin de encontrar el valor más adecuado. Se comprobó que, tanto si el valor de dicho parámetro era menor de 0.6 como mayor de 0.7, los resultados del *matching* presentaban mucho error, existiendo mucha rotación y traslación en la alineación. Por ello, se decidió seleccionar un valor intermedio. Dicho valor se fijó en $\alpha = 0.64$.

6.1 Pruebas

Una vez se han definido todos los parámetros del algoritmo, se van a describir las diferentes pruebas realizadas con sus modificaciones. Para iniciar estas pruebas, primero hay que llamar a la función *RGBD_Mapping* en la Ventana de Comandos (*Command Window*) de MATLAB. Una vez llamada, el algoritmo requiere que se le indique con qué par de *scans* se va a realizar el *matching*. A continuación, comienza la ejecución del algoritmo. A lo largo del proceso, se va mostrando la mejor solución obtenida cada cinco iteraciones.

- **Algoritmo básico**

Como ya se describió anteriormente, para poder obtener la primera población a partir de la distribución normal, es necesario calcular el valor inicial de

la media y la matriz de covarianza. En esta primera prueba, esos valores se calculan a partir de todos los individuos de la población. Los valores de las generaciones posteriores ya sí son obtenidos empleando la ponderación.

En esta prueba tampoco se implementó la condición *if* con el fin de quedarnos siempre con la mejor población, sino que el algoritmo itera hasta el número máximo de iteraciones, mostrando como mejor la última población obtenida.

Los resultados obtenidos se muestran en las siguientes figuras:

- *Figura 15* y *16*: imagen del *matching* y resultados de las iteraciones 95 y 100 entre los *scans* 50 y 60
- *Figura 17* y *18*: imagen del *matching* y resultados de las iteraciones 95 y 100 entre los *scans* 105 y 106
- *Figura 19* y *20*: imagen del *matching* y resultados de las iteraciones 95 y 100 entre los *scans* 900 y 901

Scans 50-60

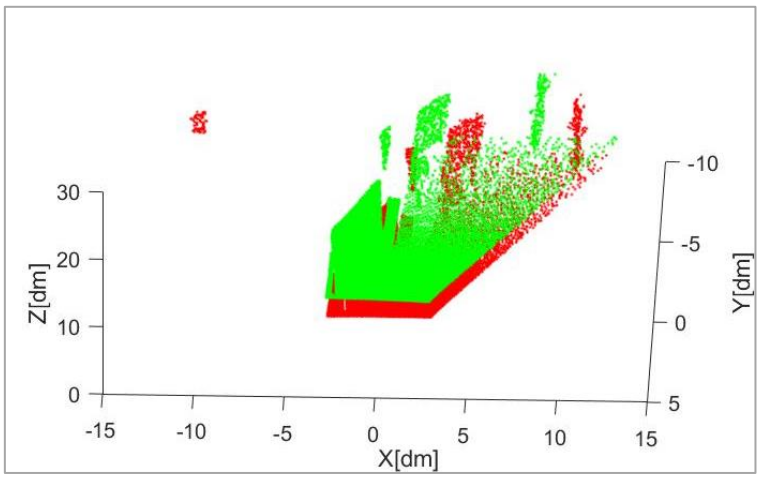

Figura 15. Matching scans 50-60 algoritmo básico

```
It: 95.000000 Best 6.342195 Worst: 220.668130 Global: 7716.041440
Position: x: 0.551022 y: -0.634921 z: 1.079638
Orientation: phi: -0.123645 theta: 0.043627 psi: 0.086207

It: 100.000000 Best 6.151477 Worst: 229.245586 Global: 7968.703884
Position: x: -0.093619 y: -1.061018 z: -0.131474
Orientation: phi: 0.008597 theta: -0.002755 psi: 0.034456
Elapsed time is 767.232210 seconds.
```

Figura 16. Resultados matching scans 50-60 algoritmo básico

Scans 105-106

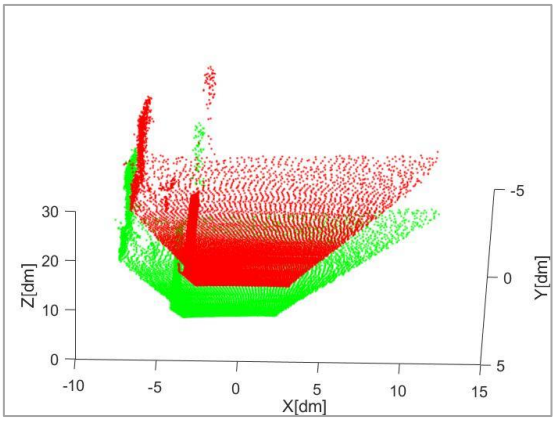

Figura 17. Matching scans 105-106 algoritmo básico

```
It: 95.000000 Best 5.223738 Worst: 120.770976 Global: 3783.621986
Position: x: -0.524395 y: -0.345602 z: 1.038577
Orientation: phi: 0.011035 theta: 0.062204 psi: 0.081875

It: 100.000000 Best 6.887682 Worst: 130.639600 Global: 4410.020752
Position: x: -0.929487 y: 1.126106 z: -0.283398
Orientation: phi: -0.090066 theta: 0.034399 psi: -0.035531
Elapsed time is 72.870341 seconds.
```

Figura 18. Resultados matching scans 105-106 algoritmo básico

Scans 900-901

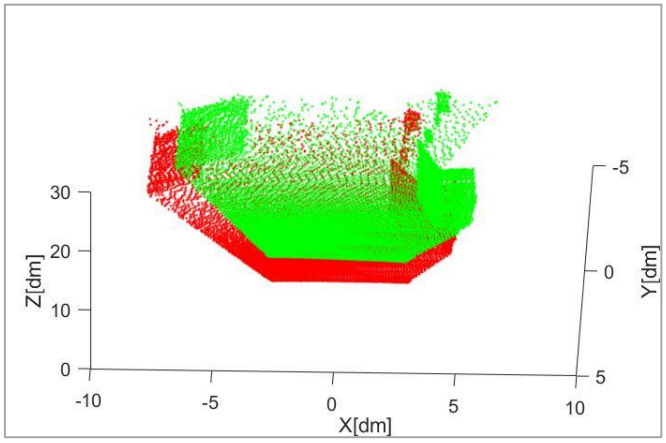

Figura 19. Matching scans 900-901 algoritmo básico

```
It: 95.000000 Best 5.494948 Worst: 135.585933 Global: 5601.365012
Position: x: -1.237908 y: 0.722535 z: -0.843887
Orientation: phi: 0.012752 theta: 0.013914 psi: 0.097307

It: 100.000000 Best 6.092044 Worst: 134.583776 Global: 5446.506773
Position: x: -0.819830 y: -1.058365 z: 0.061654
Orientation: phi: 0.000486 theta: 0.115249 psi: -0.003674
Elapsed time is 318.392100 seconds.
```

Figura 20. Resultados matching scans 900-901 algoritmo básico

Como se puede ver, el *matching* entre los tres casos presenta mucho error, tanto de rotación como de traslación.

- **Algoritmo básico con condición de *if***

En esta segunda prueba, los cálculos de la media y la matriz de covarianza no varían del caso anterior. Se añade la condición *if* mediante la que se guarda el valor de la mejor población.

Los resultados obtenidos se muestran en las siguientes figuras:

- *Figura 21* y *22*: imagen del *matching* y resultados de las iteraciones 95 y 100 entre los *scans* 50 y 60
- *Figura 23* y *24*: imagen del *matching* y resultados de las iteraciones 95 y 100 entre los *scans* 105 y 106
- *Figura 25* y *26*: imagen del *matching* y resultados de las iteraciones 95 y 100 entre los *scans* 900 y 901

Scans 50-60

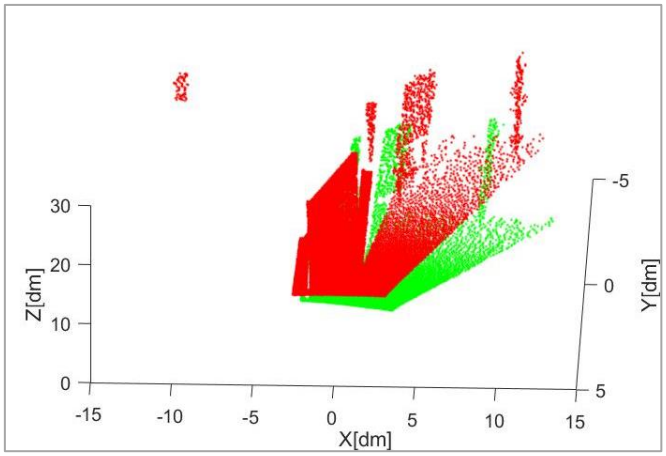

Figura 21. Matching scans 50-60 algoritmo con condición de if

```
It: 95.000000 Best 0.678584 Worst: 10.004873 Global: 365.487012
Position: x: 0.729898 y: -0.108166 z: 1.074474
Orientation: phi: -0.142757 theta: -0.002528 psi: 0.069822

It: 100.000000 Best 0.678584 Worst: 10.004873 Global: 365.487012
Position: x: 0.729898 y: -0.108166 z: 1.074474
Orientation: phi: -0.142757 theta: -0.002528 psi: 0.069822
Elapsed time is 698.560541 seconds.
```

Figura 22. Resultados matching scans 50-60 algoritmo con condición de if

Scans 105-106

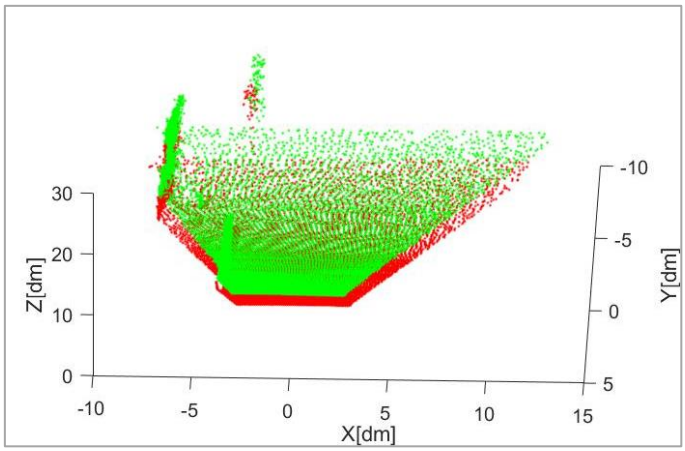

Figura 23. Matching scans 105-106 algoritmo con condición de if

```
It: 95.000000 Best 1.002532 Worst: 20.002731 Global: 650.961003
Position: x: -0.572867 y: -0.756377 z: -1.124618
Orientation: phi: 0.068624 theta: 0.042643 psi: -0.002660

It: 100.000000 Best 1.002532 Worst: 20.002731 Global: 650.961003
Position: x: -0.572867 y: -0.756377 z: -1.124618
Orientation: phi: 0.068624 theta: 0.042643 psi: -0.002660
Elapsed time is 81.675373 seconds.
```

Figura 24. Resultados matching scans 105-106 algoritmo con condición de if

Scans 900-901

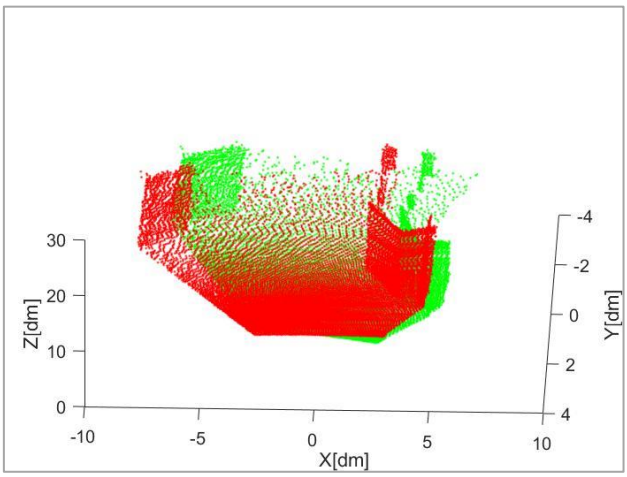

Figura 25. Matching scans 900-901 algoritmo con condición de if

```
It: 95.000000 Best 0.674372 Worst: 5.476771 Global: 265.229328
Position: x: -0.831435 y: 0.191241 z: 0.478277
Orientation: phi: 0.032277 theta: 0.133931 psi: 0.077219

It: 100.000000 Best 0.674372 Worst: 5.476771 Global: 265.229328
Position: x: -0.831435 y: 0.191241 z: 0.478277
Orientation: phi: 0.032277 theta: 0.133931 psi: 0.077219
Elapsed time is 373.170668 seconds.
```

Figura 26. Resultados matching scans 900-901 algoritmo con condición de if

Como se puede observar, estos resultados tampoco son correctos: sigue existiendo mucha rotación y traslación, por lo que el *matching* no es correcto. En alguno de los casos, sí se aproxima algo más a la solución, como en la *Figura 22*, pero hay mucha traslación en el eje *z*.

- **Algoritmo con media inicial y matriz de covarianza inicial ponderada**

Con el fin de mejorar los resultados de los casos anteriores, se pensó en utilizar el mismo método para el cálculo de la media y la matriz de covarianza iniciales. Se genera una población inicial y, para poder calcular los parámetros del algoritmo, es necesario calcular el coste y ordenar los elementos de la población inicial. En este apartado, también se realiza la selección de la mejor población. El algoritmo de este experimento es el explicado en el *Capítulo 5*.

Una modificación posterior fue que, a diferencia del caso anterior, la mejor población se "guarda" a partir de la iteración 5. Esto es porque se observó que la primera población era la mejor población que el algoritmo generaba. Al dejarlo mutar un número de iteraciones, sí se obtenían variaciones que mejoraban a las generaciones anteriores. Los resultados obtenidos se muestran en las siguientes figuras:

- *Figura 27* y *28*: imagen del *matching* y resultados de las iteraciones 95 y 100 entre los *scans* 50 y 60
- *Figura 29* y *30*: imagen del *matching* y resultados de las iteraciones 95 y 100 entre los *scans* 105 y 106
- *Figura 31* y *32*: imagen del *matching* y resultados de las iteraciones 95 y 100 entre los *scans* 900 y 901

Scans 50-60

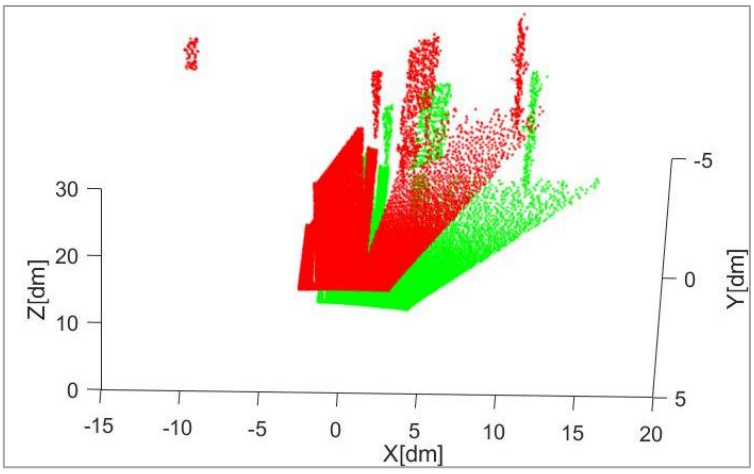

Figura 27. Matching scans 50-60 algoritmo con media y matriz ponderada

```
 It: 95.000000 Best 0.470267 Worst: 10.561367 Global: 419.487957
 Position: x: 0.778917 y: 0.343459 z: 1.078857
 Orientation: phi: -0.094976 theta: 0.103151 psi: 0.016754

 It: 100.000000 Best 0.470267 Worst: 10.561367 Global: 419.487957
 Position: x: 0.778917 y: 0.343459 z: 1.078857
 Orientation: phi: -0.094976 theta: 0.103151 psi: 0.016754
Elapsed time is 551.108952 seconds.
```

Figura 28. Resultados matching scans 50-60 algoritmo con media y matriz ponderada

Scans 105-106

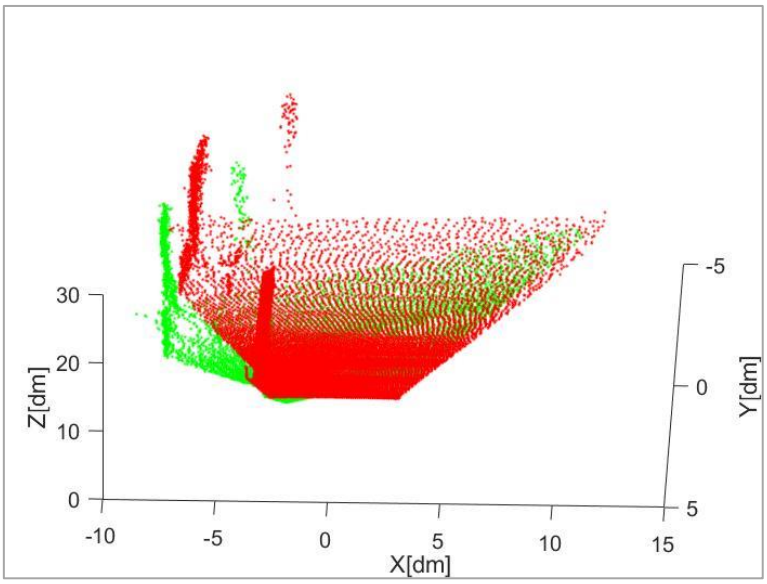

Figura 29. Matching scans 105-106 algoritmo con media y matriz ponderada

```
It: 75.000000 Best 1.086934 Worst: 11.882131 Global: 420.736033
Position: x: 0.714530 y: -1.162038 z: -0.078896
Orientation: phi: -0.135233 theta: -0.096100 psi: -0.149145

It: 80.000000 Best 1.086934 Worst: 11.882131 Global: 420.736033
Position: x: 0.714530 y: -1.162038 z: -0.078896
Orientation: phi: -0.135233 theta: -0.096100 psi: -0.149145
```

Figura 30. Resultados matching scans 105-106 algoritmo con media y matriz ponderada

Scans 900-901

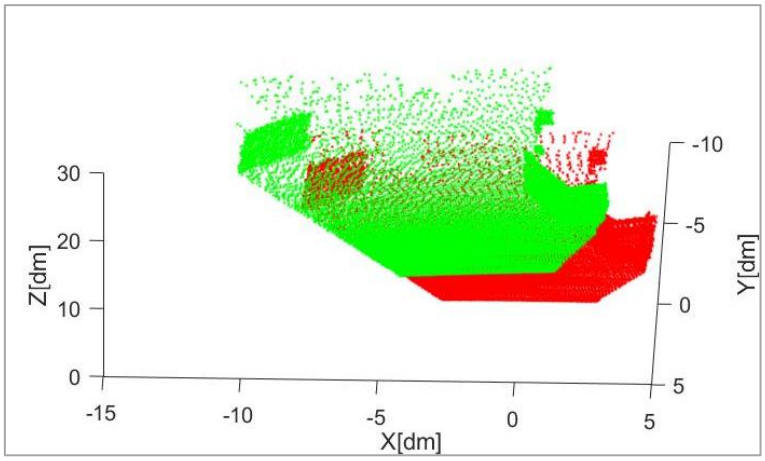

Figura 31. Matching scans 900-901 algoritmo con media y matriz ponderada

```
It: 95.000000 Best 0.430077 Worst: 8.551005 Global: 362.216776
Position: x: -1.166687 y: -0.587989 z: 0.722666
Orientation: phi: 0.119056 theta: -0.083348 psi: -0.027372

It: 100.000000 Best 0.430077 Worst: 8.551005 Global: 362.216776
Position: x: -1.166687 y: -0.587989 z: 0.722666
Orientation: phi: 0.119056 theta: -0.083348 psi: -0.027372
Elapsed time is 381.539932 seconds.
```

Figura 32. Resultados matching scans 900-901 algoritmo con media y matriz ponderada

Como se puede ver, estos resultados no son los esperados: aunque algunos de los valores sí están muy próximos a cero, otros son demasiado elevados. Si nos fijamos en la imagen anterior (*Figura 32*) se aprecia que, principalmente, los valores de x y de ϕ son más altos de lo que cabría esperar.

Si se analizan algunas de las iteraciones del algoritmo, se puede apreciar que, llegados a un determinado momento, el algoritmo no es capaz de generar mejores poblaciones. En algunos casos, desde un número de iteración bajo, el algoritmo no avanza. En la *Figura 33* se muestra un ejemplo de este fallo, que corresponde con las primeras iteraciones de la prueba realizada para este apartado con los *scans* 105-106.

```
It: 10.000000 Best 1.086934 Worst: 11.882131 Global: 420.736033
Position: x: 0.714530 y: -1.162038 z: -0.078896
Orientation: phi: -0.135233 theta: -0.096100 psi: -0.149145

It: 15.000000 Best 1.086934 Worst: 11.882131 Global: 420.736033
Position: x: 0.714530 y: -1.162038 z: -0.078896
Orientation: phi: -0.135233 theta: -0.096100 psi: -0.149145
```

Figura 33. Primeras iteraciones matching scans 105-106 algoritmo con media y matriz ponderada

Para poder comprender qué sucede, es necesario analizar la matriz de covarianza. A continuación se muestran tres tablas (*Tabla 3, Tabla 4* y *Tabla 5*), de tres iteraciones diferentes:

Iteración 1

	x	y	z	ϕ	θ	ψ
x	0,32512484	0,09830515	0,12695912	0,01335539	-0,0587448	0,01808655
y	0,09830515	0,37666781	0,28169543	0,01127156	0,06998227	0,00676627
z	0,12695912	0,28169543	0,32006532	0,03397867	0,06818272	0,00418506
ϕ	0,01335539	0,01127156	0,03397867	0,01002044	0,00293502	0,00050066
θ	-0,0587448	0,06998227	0,06818272	0,00293502	0,02046754	0,00183412
ψ	0,01808655	0,00676627	0,00418506	0,00050066	0,00183412	0,00204069

Tabla 3. Matriz de covarianza iteración 1

Iteración 2

	x	y	z	ϕ	θ	ψ
x	0,43072257	-0,13827996	0,01145009	-0,00307562	-0,06965302	0,01464268
y	-0,13827996	0,65298979	-0,12303492	-0,07927566	0,13818226	-0,02060792
z	0,01145009	-0,12303492	0,50299135	-0,00455759	-0,02770344	-0,00358129
ϕ	-0,00307562	-0,07927566	-0,00455759	0,03307098	-0,02156361	0,00964712
θ	-0,06965302	0,13818226	-0,02770344	-0,02156361	0,0387128	-0,00892886
ψ	0,01464268	-0,02060792	-0,00358129	0,00964712	-0,00892886	0,00644232

Tabla 4. Matriz de covarianza iteración 2

Iteración 10

	x	y	z	ϕ	θ	ψ
x	18,335417	9,2637426	13,149569	0,75564177	0,5899297	0,11711756
y	9,2637426	18,8818505	10,5767804	0,35271421	0,65553621	0,3315796
z	13,149569	10,5767804	21,0552276	0,45801747	0,71890377	0,47462735
ϕ	0,75564177	0,35271421	0,45801747	0,20386805	0,03119621	-0,01600065
θ	0,5899297	0,65553621	0,71890377	0,03119621	0,14980906	0,01238659
ψ	0,11711756	0,3315796	0,47462735	-0,01600065	0,01238659	0,10618948

Tabla 5. Matriz de covarianza iteración 10

Si se observan los datos obtenidos para cada iteración, se aprecia que muchos de los valores de la matriz se elevan con el número de iteraciones. Esto es porque el algoritmo está generando las nuevas poblaciones alejándose de la solución. Es decir, a mayores valores de covarianza, más lejos se encuentra de la solución.

- **Algoritmo con menor valor de media**

Al igual que la covarianza, el valor de la media de la población va aumentando, ya que los puntos están cada vez más alejados de la solución. Por ello, se va a implementar otra solución, en la que si la media de la nueva generación es mayor que la anterior, se utilice la de la población anterior para el cálculo de la distribución normal.

La *Figura 34* muestra los resultados del matching obtenidos con dos *scans* (50-55) diferentes a los usados en los casos anteriores, y en la *Figura 35* se exponen los valores de la traslación y rotación que existen tras realizar la alineación.

Scans 50-55

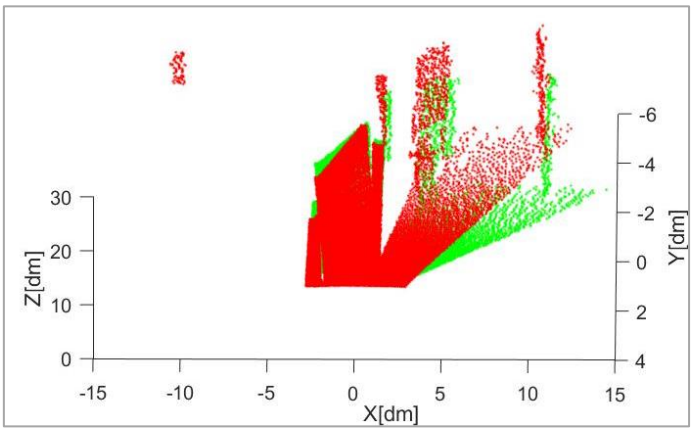

Figura 34. Matching scans 50-55 algoritmo con menor valor de media

```
It: 95.000000 Best 2.305151 Worst: 36.332774 Global: 1208.356493
Position: x: -0.340461 y: 0.707241 z: -0.429191
Orientation: phi: -0.102988 theta: 0.071845 psi: 0.039521

It: 100.000000 Best 1.584343 Worst: 27.912559 Global: 1114.423805
Position: x: -0.216177 y: -0.837039 z: 1.104935
Orientation: phi: -0.111587 theta: 0.076617 psi: 0.042047
Elapsed time is 531.208516 seconds.
```

Figura 35. Resultados matching scans 50-55 algoritmo con menor valor de media

Las *Figuras 36 y 37* representan lo mismo que las dos anteriores, salvo que se han empleado los barridos 50-60.

Scans 50-60

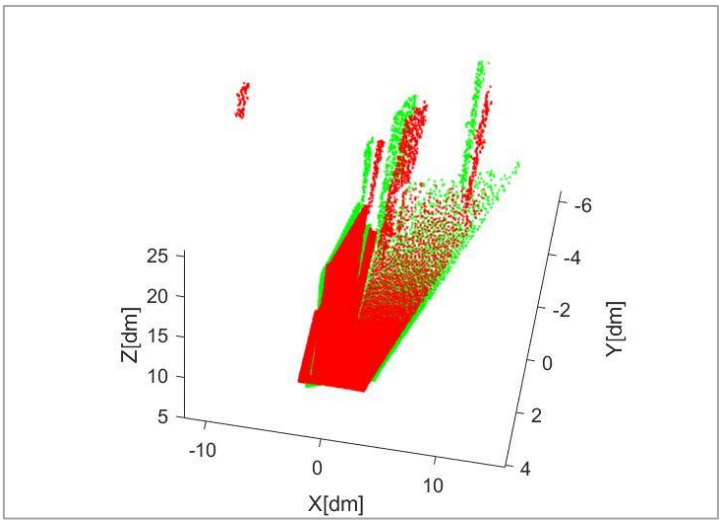

Figura 36. Matching scans 50-60 algoritmo con menor valor de media

```
 It: 95.000000 Best 2.140579 Worst: 34.911567 Global: 1361.786914
 Position: x: 0.753647 y: -1.181591 z: -0.575478
 Orientation: phi: -0.009162 theta: -0.051927 psi: 0.141595

 It: 100.000000 Best 2.398281 Worst: 41.286956 Global: 1642.117920
 Position: x: 0.198042 y: 0.074157 z: 1.231975
 Orientation: phi: -0.032396 theta: 0.065074 psi: -0.125210
Elapsed time is 554.198503 seconds.
```

Figura 37. Resultados matching scans 50-60 con menor valor de media

Como se puede ver, los resultados han mejorado, pero no tanto como se esperaba. El resultado del *matching* sigue siendo incorrecto.

Ante estos resultados, se aumenta el número de iteraciones de 100 a 200, con el fin de que el algoritmo sea capaz de llegar una solución mejor. Como se puede ver en las *Figuras 38* y *39*, los resultados obtenidos en cuanto a rotación son mucho mejores, mientras que los de traslación no han mejorado en gran medida.

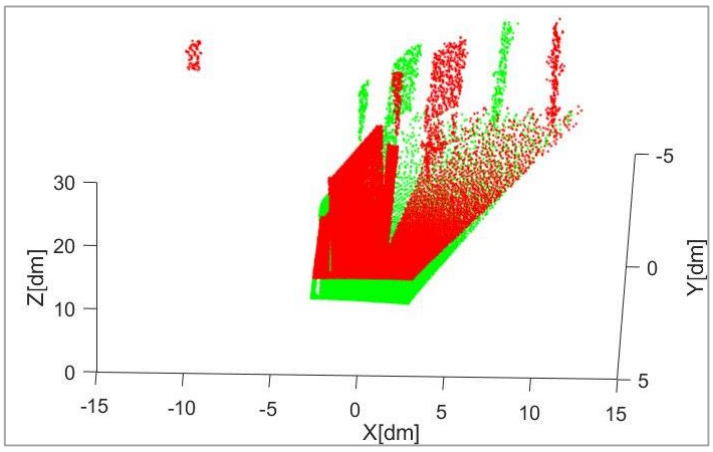

Figura 38. Matching scans 50-60 con 200 iteraciones

```
 It: 200.000000 Best 12.908327 Worst: 661.491695 Global: 21422.128006
 Position: x: 0.344273 y: 1.144235 z: -0.648232
 Orientation: phi: 0.052837 theta: -0.063281 psi: 0.051542
Elapsed time is 1864.555799 seconds.
```

Figura 39. Resultados scans 50-60 con 200 iteraciones

Dado que realizando un control sobre la media se ha obtenido una leve mejora en los resultados, se opta por realizar algún cambio en el criterio de selección.

- **Algoritmo con menor valor de media y de matriz de covarianza**

Con el fin de intentar "redirigir" las nuevas poblaciones hacia soluciones mejores, se escoge el mejor valor de matriz de covarianza entre la generación actual y la anterior, de manera que la distribución se calcule con la menor de ellas. Además, se mantiene el criterio de selección de la menor media como en el caso anterior.

Al escogerse el menor valor de la media para la distribución, también repercute en el cálculo de la matriz de covarianza, dado que se calcula con el mismo valor de media que la distribución.

En las *Figuras 40* y *41* se muestran los resultados del *matching* con los *scans* 50-60, en las que se puede ver que los resultados distan de ser correctos, aunque, como pasa en casi todos los casos, sí se consigue dar con buenos resultados para alguno de los valores.

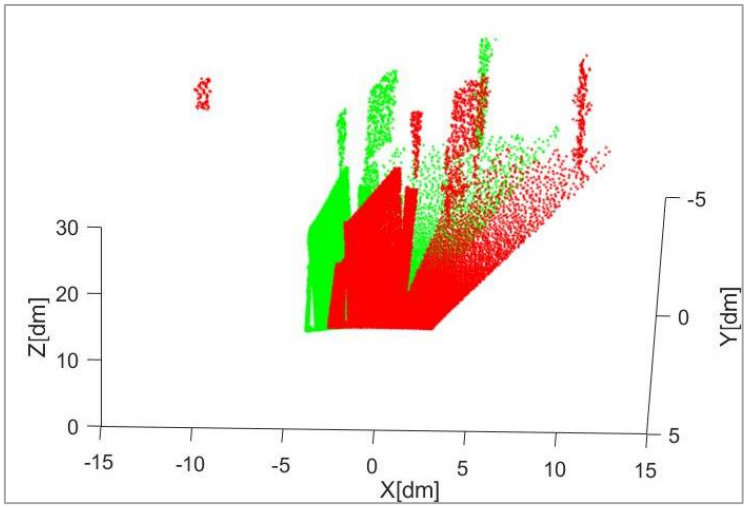

Figura 40. Matching scans 50-60 con menor media y matriz de covarianza

```
It: 90.000000 Best 0.812337 Worst: 1.298060 Global: 102.627945
Position: x: -1.367702 y: -1.232562 z: -0.696261
Orientation: phi: -0.003383 theta: -0.118958 psi: -0.067164

It: 95.000000 Best 0.854765 Worst: 1.484839 Global: 103.097877
Position: x: 1.440428 y: 0.630483 z: 1.374665
Orientation: phi: -0.115487 theta: -0.033740 psi: 0.035301

It: 100.000000 Best 0.867713 Worst: 1.257378 Global: 103.971462
Position: x: -0.793836 y: 0.012916 z: 0.334662
Orientation: phi: -0.005140 theta: -0.104848 psi: -0.038563
Elapsed time is 253.036098 seconds.
```

Figura 41. Resultados scans 50-60 con menor media y matriz de covarianza

Se ha podido comprobar que el algoritmo no tiene tendencia de disminución del valor del coste, sino que va variando entre iteraciones. Al intentar seleccionar la mejor población según el coste, se observó que la población inicial era la que presentaba un menor valor, y que el algoritmo rara vez era capaz de mejorar ese valor de coste. A pesar de todo, los valores de error de rotación y traslación en iteraciones posteriores eran mejores que los de la población inicial de menor coste.

Por ello, se introdujeron unos valores de umbral. De este modo, si el algoritmo presentaba en alguna de las iteraciones un valor menor de error de $x, y, z, \phi, \theta, \psi$ que dicho umbral, el algoritmo debía parar, mostrando dicha solución. Tras la realización de numerosas pruebas, se observó que algunos de los valores sí cumplían dichas condiciones, pero que eran pocas las veces que conseguía converger. El algoritmo se mueve cerca de la solución, pero no llega hasta ella.

En la *Figura 42* puede verse un ejemplo de esto. La iteración 40 presenta buenos resultados de error en cuatro de los valores, siento z y ψ los que se encontraban más lejos de la solución.

```
It: 35.000000 Best 0.846117 Worst: 1.496930 Global: 101.654061
Position: x: 0.973129 y: 1.447990 z: 0.690746
Orientation: phi: -0.046837 theta: 0.025221 psi: -0.117669

It: 40.000000 Best 0.813472 Worst: 1.311249 Global: 99.909126
Position: x: 0.000067 y: -0.060234 z: 1.214167
Orientation: phi: 0.032960 theta: 0.035300 psi: 0.107833

It: 45.000000 Best 0.855289 Worst: 1.353654 Global: 101.570181
Position: x: 0.836407 y: -0.229641 z: -1.227530
Orientation: phi: -0.070059 theta: -0.103903 psi: -0.065698
```

Figura 42. Resultados iteraciones intermedias

- **Implementación del método *Rank-μ-Update***

El último experimento realizado es la implementación del *Rank-μ-Update*, explicado en el *Capítulo 4*. En este caso, la matriz de covarianza se calcula teniendo en cuenta el valor de la matriz de la generación anterior, de manera que la expresión implementada para el cálculo es la siguiente:

$$C^{(g+1)} = \left(1 - c_\mu\right) * C^{(g)} + c_\mu * C_{CMA-ES}^{(g+1)} \quad (6.1)$$

Donde:

$C^{(g+1)}$ corresponde con el valor de la matriz de covarianza utilizada para calcular la distribución;

c_μ valor empleado para dar una ponderación a las matrices según la generación. Se calcula como $c_\mu = min\left\{1, \frac{\lambda/4}{n^2}\right\}$;

$C_{CMA-ES}^{(g+1)}$ es la matriz de covarianza calculada según la *Ecuación 4.11.*

Este método está pensado para tamaños de población menores. De este modo, se consigue un menor valor de c_μ, y se le da más importancia a la generación nueva en el cálculo. Por ello, se disminuye el tamaño de la población de 100 a 50 individuos.

Se realizaron diversas pruebas con este método. Algunos de los resultados obtenidos se muestran en las siguientes figuras: *Figuras 43* y *44* utilizando los *scans* 50-51, *Figuras 45* y *46* utilizando los *scans* 900-901.

Scans 50-51

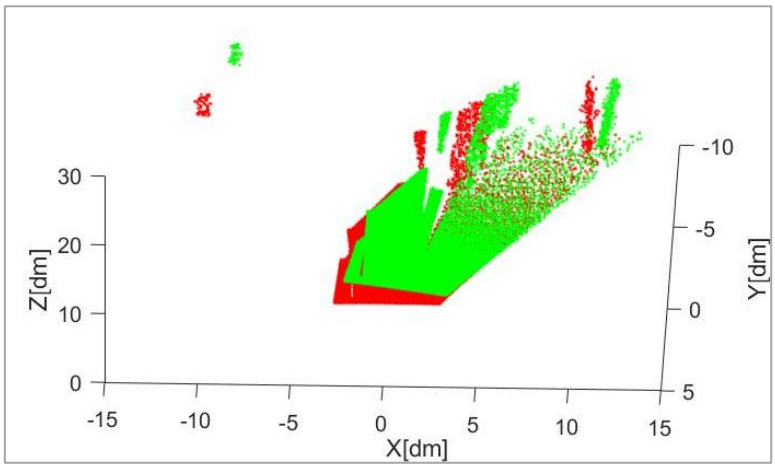

Figura 43. Matching scans 50-51 método Rank-μ-Update

```
 It: 95.000000 Best 8.416120 Worst: 140.433386 Global: 2292.656308
 Position: x: 1.354169 y: -1.430652 z: -0.113220
 Orientation: phi: 0.062632 theta: 0.114738 psi: -0.111712

 It: 100.000000 Best 4.108498 Worst: 109.000742 Global: 2144.148743
 Position: x: 0.589986 y: -0.767529 z: -0.551308
 Orientation: phi: 0.062189 theta: 0.008788 psi: 0.144015
Elapsed time is 357.199268 seconds.
```

Figura 44. Resultados scans 50-51 método Rank-μ-Update

Scans 900-901

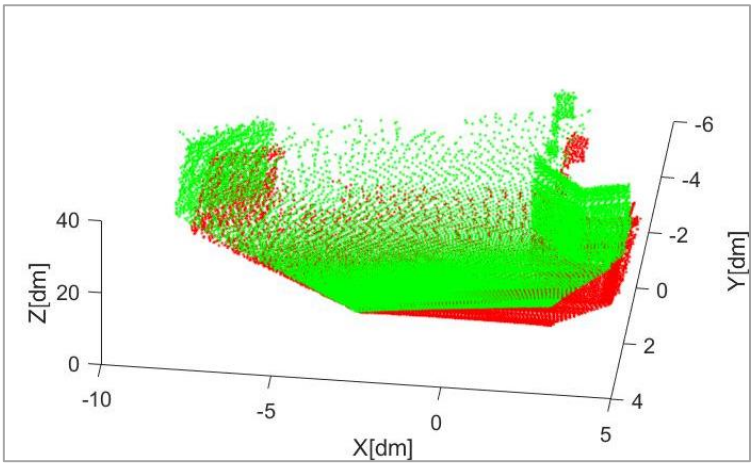

Figura 45. Matching scans 900-901 método Rank-μ-Update

```
 It: 95.000000 Best 6.367375 Worst: 220.736875 Global: 3703.304101
 Position: x: -0.182421 y: -0.366725 z: 0.561590
 Orientation: phi: 0.046054 theta: -0.070526 psi: 0.020340

 It: 100.000000 Best 8.096577 Worst: 230.440172 Global: 3497.476171
 Position: x: -0.102933 y: 0.318626 z: -0.069193
 Orientation: phi: 0.117538 theta: -0.029259 psi: -0.110704
Elapsed time is 151.539213 seconds.
```

Figura 46. Resultados scans 900-901 método Rank-μ-Update

Como se puede apreciar, los resultados no difieren demasiado de los obtenidos para casos anteriores.

Capítulo 7.

Conclusiones

Tras la realización del proyecto en su conjunto, implementación y desarrollo del algoritmo y redacción de la memoria, se ha llegado a las conclusiones que se exponen en este capítulo.

7.1 Conclusiones sobre los resultados

Tras la realización de numerosas pruebas tanto con el algoritmo implementado, como con las diferentes posibles mejoras planteadas, se puede concluir que se obtienen resultados aceptables en un número pequeño de casos. En prácticamente todas las pruebas, cuatro de los seis elementos que caracterizan a la posible solución se pueden considerar buenos resultados, siendo el quinto y sexto los que se alejaban más de la solución. El algoritmo se mueve cerca de la solución, pero no llega a converger a la óptima.

Si estos resultados se comparan con el trabajo en que se basa este proyecto, se puede concluir que el algoritmo de *Scan Matching* basado en la estrategia evolutiva *Differential Evolution* (DE) implementado por Martin *et al.* [4] proporciona mejores resultados al realizar la alineación entre dos barridos utilizando las propiedades del color.

Esto puede deberse a diferentes motivos. El primero es que, aunque ambos forman parte del conjunto de las ES, tienen un funcionamiento diferente. De este modo, es probable que el DE se adapte mejor al problema. Hay que tener en cuenta que cuando se quiere buscar solución a un problema real, se debe valorar cómo se van a seleccionar los datos, así como la manera en la que se quieren tratar, y con ello, analizar y buscar qué método puede adaptarse mejor a dicho problema. Por este motivo, existe la posibilidad de que el CMA-ES no sea un método óptimo para encontrar la solución al *matching* utilizando las propiedades del color para la extracción de los puntos característicos.

Otro motivo por el cual el algoritmo no haya alcanzado los resultados esperados puede haber sido el propio desarrollo del mismo debido a una falta de conocimiento profundo en la base de la materia tratada en el trabajo. A pesar de estas limitaciones, la realización de este proyecto permitiría continuar la

investigación del mismo, teniendo una mejor idea de los fallos que presenta el planteamiento inicial.

Otra de las razones por las que el *matching* no sea correcto puede residir en la forma en la que se calcula la distribución normal, ya que se observa que las nuevas poblaciones se generan alejándose de la solución. Para obtener la distribución, se utiliza una función propia de MATLAB, *mvnrnd*. Esta función devuelve una matriz de N_p vectores aleatorios escogidos de la distribución normal multivariante de media m y matriz de covarianza C.

7.2 Posibles líneas de mejora

Como ya se ha explicado, analizando los resultados se ha podido ver que la media de la población y la matriz de covarianza iban aumentando al incrementarse el número de iteraciones. Esto es porque el algoritmo está generando las posibles soluciones alejándose de la solución óptima.

Con algunas de las mejoras implantadas, tales como escoger el menor valor de la media, disminuir el tamaño de la covarianza o aumentar el número de iteraciones, se han podido obtener unos resultados más adecuados que para los casos anteriores, pudiendo indicar que se debería explotar esa línea de investigación. A pesar de ello, siguen sin ser los esperados.

Las posibles líneas futuras a desarrollar tras este proyecto irían dirigidas a controlar los valores de la covarianza, de manera que los datos se modificasen para ir dirigiendo al algoritmo hacia la solución óptima, estudiando más en detalle las posibles variaciones del método CMA-ES en que se basa este trabajo [49]. Otra posibilidad a estudiar es buscar otro método para obtener la distribución normal.

Aunque se han planteado vías de investigación para continuar el estudio del problema, hay que tener en cuenta que tampoco se asegura la obtención de resultados óptimos. Como ya se ha mencionado anteriormente, no todos los métodos de optimización son útiles para resolver el mismo tipo de problemas. Dependerá de las características del problema, del tipo de datos, la forma de tratarlos, etc. Es por ello que, aunque es necesario verificar y revisar cada planteamiento, existe la posibilidad de que este enfoque no sea útil a la hora de realizar la alineación entre dos barridos utilizando las propiedades del color.

7.3 Valoración general del proyecto

Como conclusión final del proyecto, a pesar de no haber conseguido unos resultados óptimos a nivel técnico, a nivel personal ha resultado ser muy satisfactorio, ya que me he tenido que enfrentar a un problema real, con una serie de dificultades no esperadas. Estas dificultades me han hecho realizar un estudio de la base del problema en profundidad, así como un análisis de las alternativas el planteamiento inicial a través de estudios en bibliografía. Esto me ha permitido conseguir un conocimiento más profundo e interés sobre la materia.

Bibliografía

[1] Proyectosalonhogar.com. (2016). *Historia de los Mapas*. [online] [Acceso 6 Sep. 2016]. Disponible en: http://www.proyectosalonhogar.com/el_porque_de_las_cosas/historia_de_los_mapas.htm

[2] BEEVERS, Kristopher R. *Modern Robot Mapping Techniques*. 2006.

[3] HENRY, Peter, *et al.* RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, 2012, vol. 31, no 5, p. 647-663.

[4] MARTÍN, Fernando; MIRÓ, Jaime Valls; MORENO, Luis. RGB-D DE-based Scan Matching: Exploiting Colour Properties in Registration. *Journal of Intelligent & Robotic Systems*, 2015, vol. 80, no 1, p. 71-85.

[5] MARTÍNEZ, Jorge L., *et al.* Mobile robot motion estimation by 2D scan matching with genetic and iterative closest point algorithms. *Journal of Field Robotics*, 2006, vol. 23, no 1, p. 21-34.

[6] RAMOS, Fabio T.; FOX, Dieter; DURRANT-WHYTE, Hugh F. CRF-Matching: Conditional Random Fields for Feature-Based Scan Matching. *Robotics: Science and Systems*. 2007

[7] THRUN, Sebastian; BURGARD, Wolfram; FOX, Dieter. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. En*Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*. IEEE, 2000. p. 321-328.

[8] POMERLEAU, François; STEINMANN, Simon. Analysis of Scan Matching Methods for Indoor 3D Mapping. *Journal of Field Robotics*, 2007, vol. 24, no 10, p. 803-827.

[9] LU, Feng; MILIOS, Evangelos. Robot pose estimation in unknown environments by matching 2D range scans. *Journal of Intelligent and Robotic Systems*, 1997, vol. 18, no 3, p. 249-275.

[10] TOMONO, Masahiro. A scan matching method using euclidean invariant signature for global localization and map building. *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on. IEEE*, 2004. p. 866-871.

[11] DIOSI, Albert; KLEEMAN, Lindsay. Fast laser scan matching using polar coordinates. *The International Journal of Robotics Research*, 2007, vol. 26, no 10, p. 1125-1153.

[12] BESL, Paul J.; MCKAY, Neil D. Method for registration of 3-D shapes. *Robotics-DL tentative. International Society for Optics and Photonics*, 1992. p. 586-606

[13] POMERLEAU, Francois; OSWALD, Lorenz. Recent Development of the Iterative Closest Point Algorithm. An overview of the years 2008 to 2010, 2010

[14] RUSINKIEWICZ, Szymon; LEVOY, Marc. Efficient variants of the ICP algorithm. *3-D Digital Imaging and Modeling, 2001. Proceedings. Third International Conference on*. IEEE, 2001. p. 145-152.

[15] JOHNSON, Andrew Edie; KANG, Sing Bing. Registration and integration of textured 3D data. *Image and vision computing*, 1999, vol. 17, no 2, p. 135-147.

[16] YOSHITAKA, Hara, *et al.* Mobile robot localization and mapping by scan matching using laser reflection intensity of the sokuiki sensor. *IECON 2006-32nd Annual Conference on IEEE Industrial Electronics*. IEEE, 2006. p. 3018-3023.

[17] CENSI, Andrea. An ICP variant using a point-to-line metric. *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on. IEEE*, 2008. p. 19-25.

[18] BONACCORSO, Filippo; MUSCATO, Giovanni; BAGLIO, Salvatore. Laser range data scan-matching algorithm for mobile robot indoor self-localization. *World Automation Congress (WAC), 2012. IEEE*, 2012. p. 1-5.

[19] NÜCHTER, Andreas, *et al.* 6D SLAM—3D mapping outdoor environments. *Journal of Field Robotics*, 2007, vol. 24, no 8- 9, p. 699-722.

[20] MALLIOS, Angelos, *et al.* Scan matching SLAM in underwater environments. Autonomous Robots, 2014, vol. 36, no 3, p. 181-198.

[21] BOSSE, Michael, *et al.* Simultaneous localization and map building in large-scale cyclic environments using the Atlas framework. *The International Journal of Robotics Research*, 2004, vol. 23, no 12, p. 1113-1139.

[22] THRUN, Sebastian; BURGARD, Wolfram; FOX, Dieter. A real-time algorithm for mobile robot mapping with applications to multi-robot and 3D mapping. *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on. IEEE*, 2000. p. 321-328.

[23] SCHADLER, Mark; STÜCKLER, Jörg; BEHNKE, Sven. Rough terrain 3D mapping and navigation using a continuously rotating 2D laser scanner. *KI-Künstliche Intelligenz*, 2014, vol. 28, no 2, p. 93-99.

[24] MAY, Stefan, *et al.* Robust 3D-mapping with time-of-flight cameras. *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009. p. 1673-1678.

[25] HENRY, Peter, *et al.* RGB-D mapping: Using Kinect-style depth cameras for dense 3D modeling of indoor environments. *The International Journal of Robotics Research*, 2012, vol. 31, no 5, p. 647-663.

[26] MENEGATTI, Emanuele, *et al.* Omnidirectional vision scan matching for robot localization in dynamic environments. *IEEE transactions on robotics*, 2006, vol. 22, no 3, p. 523-535.

[27] NEWMAN, Paul; COLE, David; HO, Kin. Outdoor SLAM using visual appearance and laser ranging. *Proceedings 2006 IEEE International Conference on Robotics and Automation,* 2006. ICRA 2006. IEEE, 2006. p. 1180-1187.

[28] KHOSHELHAM, Kourosh; ELBERINK, Sander Oude. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors,* 2012, vol. 12, no 2, p. 1437-1454.

[29] NEWCOMBE, Richard A., *et al.* KinectFusion: Real-time dense surface mapping and tracking. *Mixed and augmented reality (ISMAR), 2011 10th IEEE international symposium on.* IEEE, 2011. p. 127-136.

[30] BORRMANN, Dorit, et al. Globally consistent 3D mapping with scan matching. *Robotics and Autonomous Systems*, 2008, vol. 56, no 2, p. 130-142.

[31] MONTOYO BOJO, Javier. Estudio y mejora de métodos de registro 3D: aceleración sobre unidades de procesamiento gráfico y caracterización del espacio de transformaciones iniciales. 2015.

[32] MINGUEZ, Javier; LAMIRAUX, Florent; MONTESANO, Luis. Metric-based scan matching algorithms for mobile robot displacement estimation. *IEEE International Conference on Robotics and Automation.* IEEE; 1999, 2005. p. 3557.

[33] BIBER, Peter; STRAßER, Wolfgang. The normal distributions transform: A new approach to laser scan matching. *Intelligent Robots and Systems, 2003. (IROS 2003). Proceedings. 2003 IEEE/RSJ International Conference on.* IEEE, 2003. p. 2743-2748.

[34] Roldigital.es. (2016). *El modo de color RGB* | Blog Rol Digital. [online] [Acceso 20 Sep. 2016]. Disponible en: http://roldigital.es/blog/el-modo-de-color-rgb/

[35] (2016). *Entendiendo El Espacio de Color CIE L*A*B**. [online] [Acceso 15 Sep. 2016]. Konica Minolta Color, Light, and Display Measuring Instruments. Disponible en: http://sensing.konicaminolta.com.mx/2014/09/entendiendo-el-espacio-de-color-cie-lab/

[36] QUINTERO, Luis Vicente Santana; COELLO, Carlos A. Coello. Una introducción a la computación evolutiva y alguna de sus aplicaciones en Economía y Finanzas. *Revista de métodos cuantitativos para la economía y la empresa*, 2006, no 2, p. 3-26.

[37] OCHOA, Gabriela. Introducción a la Computación Evolutiva y la Morfogénesis Artificial.

[38] Escritura.proyectolatin.org. (2016). Capítulo: Conceptos-Básicos-De-Algoritmos-Evolutivos / Inteligencia Artificial. [online] [Acceso 15 Sep. 2016].Disponible en: http://escritura.proyectolatin.org/inteligencia-artificial/conceptois-basicos-de-algoritmos-evolutivos/

[39] CAPARRINI, F. and WORK, W. (2016). *Algoritmos Genéticos y Computación Evolutiva - Fernando Sancho Caparrini.* [online] [Acceso 15 Sep. 2016]. Cs.us.es. Disponible en: http://www.cs.us.es/~fsancho/?e=65

[40] *Algoritmos Genéticos*. [online] [Acceso 18 Sep. 2016]. Disponible en: http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/temageneticos.pdf

[41] BEYER, Hans-Georg; SCHWEFEL, Hans-Paul. Evolution strategies–A comprehensive introduction. *Natural computing*, 2002, vol. 1, no 1, p. 3-52.

[42] ISASI, P.. *Computación biológica*. [pdf] [Acceso 18 Sep. 2016]. Disponible en: http://ocw.uc3m.es/ingenieria-informatica/computacion-biologica/material-de-clase-1/EEvolutivasT.pdf

[43] HANSEN, Nikolaus; ARNOLD, Dirk V.; AUGER, Anne. Evolution Strategies. 2013.

[44] HORRA, J. de la. *ESTADÍSTICA DESCRIPTIVA: DOS VARIABLES*. [pdf] [Acceso 19 Sep. 2016]. Disponible en: https://www.uam.es/personal_pdi /ciencias/horra/ Estadistica-Apuntes/Descriptiva-Dos-Variables.pdf.

[45] MARIN DIAZARAQUE, Juan Miguel. *Tema 2. Estadística descriptiva multivariante*. [pdf] [Acceso 19 Sep. 2016]. Disponible en: http://halweb.uc3m.es/esp/ Personal/personas/jmmarin/esp/AMult/tema2am.pdf.

[46] HANSEN, Nikolaus; OSTERMEIER, Andreas. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. *Evolutionary Computation, 1996., Proceedings of IEEE International Conference*. IEEE, 1996. p. 312-317.

[47] HANSEN, Nikolaus; OSTERMEIER, Andreas. Convergence properties of evolution strategies with the derandomized covariance matrix adaptation: The $(\mu/\mu_I,\lambda)$-CMA-ES. *Eufit,* 1997, vol. 97, p. 650-654.

[48] HANSEN, Nikolaus; KERN, Stefan. Evaluating the CMA evolution strategy on multimodal test functions. *International Conference on Parallel Problem Solving from Nature*. Springer Berlin Heidelberg, 2004. p. 282-291.

[49] HANSEN, Nikolaus. The CMA evolution strategy: a tutorial, 2011.

Anexo

I. Presupuesto

En este capítulo, se establece el coste total de realización de este proyecto en un entorno real de trabajo. Se va a dividir en dos partes: costes de material y costes de personal. Finalmente, se muestran los costes totales del proyecto.

- **Costes de material**

Entre los costes de material se encuentra la licencia del programa empleado para implementar el algoritmo MATLAB. Existen diferentes opciones, que se detallan en la *Tabla 6.*

Tipo de licencia	Estándar	Educación	Personal	Estudiante
Precio licencia	2000€	500€	105€	35€

Tabla 6. Precio licencias MATLAB

Para este caso, el precio de la licencia es 35€.

En la *Tabla 7* se pueden observar los costes de material empleados para el proyecto.

Concepto	Coste (€)
Ordenador	600
Licencias	35
(Otros: internet, material de oficina)	100
TOTAL	**735**

Tabla 7. Costes de material

- **Costes de personal**

Los costes de personal se calcularán en función de las horas empleadas en el desarrollo del trabajo. Las horas empleadas por el alumno son pagadas a 10€, mientras que las del tutor a 20€.

En la *Tabla 8* se muestran las horas empleadas por el tutor, así como el coste.

	Tiempo empleado (h)	Coste (€)
Tutor TOTAL	40	**800**

Tabla 8. Horas empleadas por el tutor y coste

En la *Tabla 9* se muestran las horas empleadas por el alumno en cada tarea, y el coste de realización de cada una de ellas. La descripción de cada tarea se describe a continuación:

- **Tarea 1**: comprende las tareas de familiarización, búsqueda de información y análisis del funcionamiento del trabajo en que se basa este proyecto [4].
- **Tarea 2**: representa el diseño del algoritmo.
- **Tarea 3**: desarrollo del algoritmo, que incluye: familiarización con la herramienta MATLAB, implementación, pruebas, análisis, reconocimiento de fallos y búsqueda de posibles mejoras.
- **Tarea 4**: representa la redacción de la memoria, incluyendo la corrección de la misma.

Tarea	Tiempo empleado (h)	Coste (€)
1	70	700
2	12	120
3	170	1700
4	100	1000
TOTAL	**352**	**3520**

Tabla 9. Horas empleadas por el alumno y coste de cada tarea

- **Costes totales**

En la *Tabla 10* se muestran los costes totales finales de realización del proyecto, ascendiendo a 5055 €.

Tipo de coste	Coste total (€)
Material	735
Estudiante	3520
Tutor	800
TOTAL	**5055**

Tabla 10. Costes totales

II. Planificación

La planificación inicial del proyecto fue la siguiente, y puede verse reflejada en la *Figura 47*:

- La **Tarea 1**, que comprende las tareas de familiarización, búsqueda de información y análisis del funcionamiento del trabajo en que se basa este proyecto, estaba planificada para iniciarse el día 1 de febrero, y finalizar el día 15 de marzo.

- La **Tarea 2**, que representa el diseño del algoritmo, debía iniciarse al acabar a etapa anterior.

- La **Tarea 3** es la etapa de desarrollo del algoritmo, que incluye: familiarización con la herramienta MATLAB, implementación, pruebas, análisis, reconocimiento de fallos y búsqueda de posibles mejoras. Esta debía comenzar con la familiarización con la herramienta MATLAB al finalizar la Tarea 1, y terminar la primera semana de junio.

- La **Tarea 4** equivale a la fase de redacción de la memoria, incluyendo la corrección de la misma. Debía abarcar el periodo comprendido entre el mes de mayo y la tercera semana de junio.

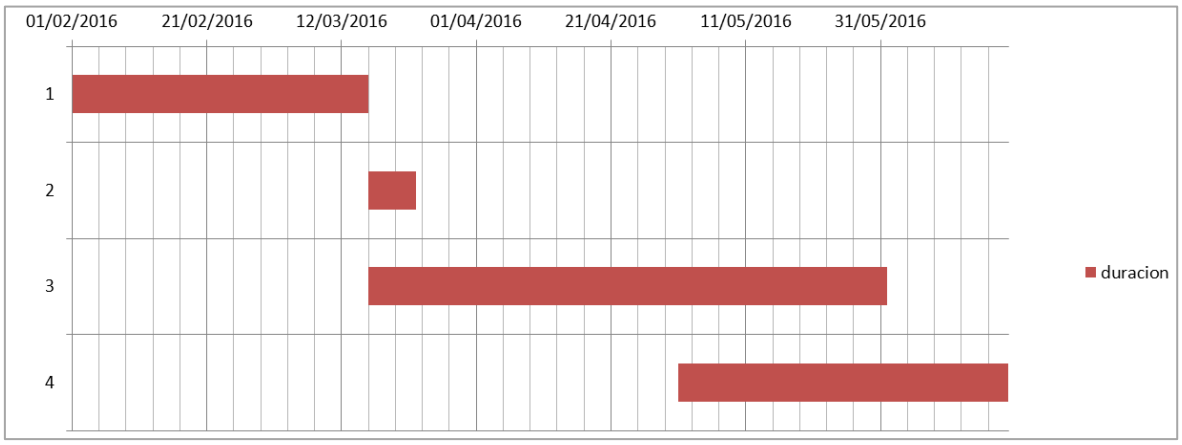

Figura 47. Diagrama de Gantt de las tareas. Planificación inicial.

Esta planificación no pudo cumplirse por diversas tareas críticas encontradas a lo largo del desarrollo del proyecto. La primera de ellas fue la dificultad de comprensión del tema y análisis de trabajo en que se basa este proyecto, lo que requirió más tiempo del planificado inicialmente.

La segunda tarea crítica surgió al intentar realizar pruebas con el trabajo en que se basa el proyecto, ya que saltaba un error que llevaba a cerrar el programa. Este error provenía de un archivo tipo .MEX. Una vez se detectó el error, se sustituyó esa función externa por una función de MATLAB que permitió continuar.

Por último, la tercera tarea crítica fue que el algoritmo implementado no realizaba el *matching* de la manera esperada, por lo que hubo que investigar y aplicar diversas modificaciones con el fin de conseguir los resultados esperados.